ELSEVIER

Computational Geometry 23 (2002) 117–151

Computational Geometry
Theory and Applications

www.elsevier.com/locate/comgeo

A decomposition-based approach to layered manufacturing ☆

Ivaylo Ilinkin [a,1], Ravi Janardan [a,*,1,2], Jayanth Majhi [b,2], Jörg Schwerdt [c,3], Michiel Smid [d,3], Ram Sriram [e]

[a] *Department of Computer Science & Engineering, University of Minnesota, Minneapolis, MN 55455, USA*
[b] *Synopsys Corporation, 700 East Middlefield Road, Mountain View, CA 94043, USA*
[c] *Algorithmic Solutions Software GmbH, Postfach 15 11 01, 66041 Saarbrücken, Germany*
[d] *School of Computer Science, Carleton University, Ottawa, ON, Canada, K15 5B6*
[e] *National Institute of Standards and Technology, Gaithersburg, MD 20899, USA*

Received 23 February 2001; received in revised form 30 July 2001; accepted 4 September 2001

Communicated by K. Mehlhorn

Abstract

This paper introduces a new approach for improving the performance and versatility of *Layered Manufacturing* (LM), which is an emerging technology that makes it possible to build physical prototypes of 3D parts directly from their computer models using a "3D printer" attached to a personal computer.

Current LM processes work by viewing the computer model as a single, monolithic unit. By contrast, the approach proposed here decomposes the model into a small number of pieces, by intersecting it with a suitably chosen plane, builds each piece separately using LM, and then glues the pieces together to obtain the physical

☆ A preliminary version of this paper appears in the Proceedings of the Seventh International Workshop on Algorithms and Data Structures, Providence, RI, 8–10 August 2001, LNCS 2125, pp. 389–400.

* Corresponding author.
E-mail addresses: ilinkin@cs.umn.edu (I. Ilinkin), janardan@cs.umn.edu (R. Janardan), majhi@synopsys.com (J. Majhi), joerg.schwerdt@algorithmic-solutions.com (J. Schwerdt), michiel@scs.carleton.ca (M. Smid), sriram@cme.nist.gov (R. Sriram).

[1] Research supported, in part, by National Science Foundation grant CCR-9712226 and by National Institute of Standards and Technology grant 60NANB8D0002. Commercial equipment and software, if any, are identified only in order to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

[2] Portions of this work were done when RJ visited the University of Magdeburg, Germany under a joint grant from the National Science Foundation and Deutscher Akademischer Austauschdienst for international research.

[3] Work done while at the Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, 0-39106 Magdeburg, Germany. Portions of this work done while visiting the University of Minnesota, under a joint grant from DAAD and NSF for international research.

PII: S0925-7721(01)00059-1

prototype. This approach allows large models to be built quickly in parallel. Furthermore, it is very efficient in its use of so-called support structures that are generated by the LM process.

This paper presents the first provably correct and efficient geometric algorithms to decompose polyhedral models so that the support requirements (support volume and area of contact) are minimized. Algorithms based on the plane-sweep paradigm are first given for convex polyhedra. These algorithms run in $O(n \log n)$ time for n-vertex convex polyhedra and work by generating expressions for the support volume and contact-area as a function of the height of the sweep plane, and optimizing them during the sweep. These algorithms are then generalized to non-convex polyhedra, which are considerably more difficult due to the complex structure of the supports. It is shown that, surprisingly, non-convex polyhedra can be handled by first identifying certain critical facets using a technique called cylindrical decomposition, and then applying the algorithm for convex polyhedra to these critical facets. The resulting algorithms run in $O(n^2 \log n)$ time. Also given is a method for controlling the size of the decomposition, so that the number of pieces generated is within a user-specified limit. Experimental results show that the proposed approach can achieve significant reduction in support requirements in both the convex and the non-convex case.

Keywords: Computational geometry; Cylindrical decomposition; Layered manufacturing; Optimization; Plane sweep; Rapid prototyping

1. Introduction

The field of computer-aided design and manufacturing has now progressed to the point where an engineer can not only design and test a virtual model of a 3D part on a personal computer, but can also generate directly from the model a physical prototype of the part, using a relatively small and inexpensive "3D printer" attached to the computer [19]. This technology, called *Rapid Prototyping* (RP), provides the engineer with an additional level of physical verification that makes it possible to detect and correct flaws that may, otherwise, have gone unnoticed. RP is used extensively in the automotive, aerospace, and medical industries.

At the heart of RP is a manufacturing technique called *Layered Manufacturing* (LM). The basic principle underlying LM is simple: the computer model is oriented suitably and sliced into thin layers by horizontal planes. The layers are then sent over a network to a fabrication device which "prints" them one by one, each on top of the previous one.

Fig. 1 depicts a widely-used LM process called Stereolithography. In essence, the Stereolithography Apparatus (SLA) consists of a vat of light-sensitive liquid resin, a platform, and a laser. The input to the process (and to virtually all other LM processes) is a surface triangulation of the model in the industry-standard STL format. This format merely consists of an unordered list of the triangles, where each triangle is specified by listing its three vertices and its outward-directed unit-normal.[4] The model is oriented suitably, sliced by horizontal planes, and then built in the vertical direction as follows: Initially, the platform is below the surface of the resin at a depth equal to the layer thickness. The laser traces out the contour of the first slice on the surface and then fills in the interior in a raster-like pattern – a process called *hatching*. The resin then hardens to a depth equal to the slice thickness and forms the first layer, which rests on the platform. Next, the platform is lowered by an amount equal to the layer thickness, the

[4] The simplicity of the STL format has led to its widespread adoption in the LM industry. However, the format does have some inadequacies and alternative formats have been proposed; see, for instance, [15,26].

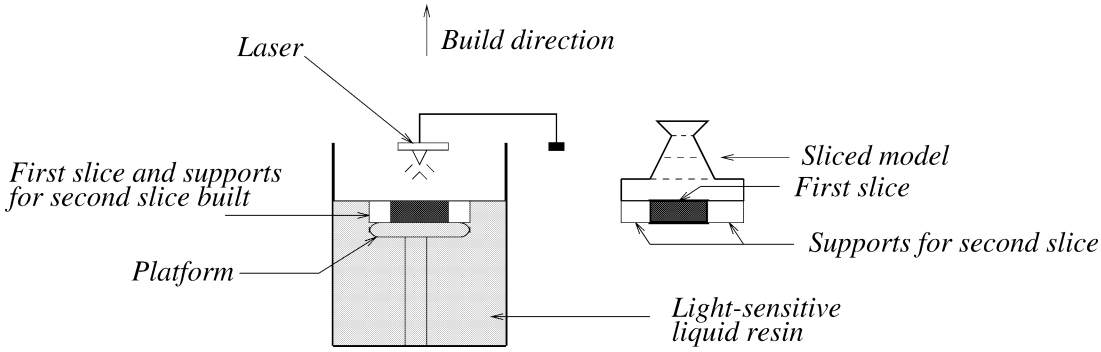

Fig. 1. The stereolithography apparatus.

vacated region is re-coated with resin, and the second layer is then built in the same way. Ideally, each layer after the first one should rest in its entirety on the previous one. In general, however, portions of a layer can overhang the previous layer, so additional structures, called *supports*, are needed to hold up the overhangs. Supports are generated automatically during the process itself. For this the model is analyzed a priori and a suitable description of the supports is generated and merged into the STL file. Once the part has been made, it is postprocessed to remove the supports and to improve the surface finish, which has a stair-stepped appearance owing to the discretization into layers.

1.1. Geometric issues in layered manufacturing

Unfortunately, the time requirements of present-day LM processes is quite high – often running into hours. Generally, an LM process consists of three phases: preprocessing, building, and postprocessing. Preprocessing includes repairing flaws in the STL file (e.g., gaps between facets, geometric singularities, etc. [5,8]), deciding upon a suitable initial orientation for the model (or, equivalently, the *build direction*), computing support requirements, generating and merging a description of the supports into the STL file, and slicing the model and supports. The building phase involves tracing and hatching each layer. Postprocessing includes removal of supports and improving part finish and accuracy.

All of these tasks are influenced by geometric considerations [5,17]. In particular, the orientation of the part impacts the number of layers, the quantity of supports, the location of supports on the part, the extent to which supports "stick" to the part, and the surface finish and accuracy. The layer geometry affects the tool-path during hatching. The STL representation is not well suited for the efficient computation of slices and supports as it does not contain any topological information about the model.

In current systems these issues are resolved in an ad hoc manner, through human intervention. For instance, the part is oriented by the operator, based on experience, so that the quantity of supports used is "small" and the finish is "good". However, over the past few years the problem of automating these decisions via efficient and correct algorithms has been addressed by several researchers in computational geometry.

Asberg et al. [3] (see also [9]), describe efficient algorithms to decide if a given model can be built without supports using Stereolithography. Majhi et al. [25] give algorithms to minimize the volume of supports and the area of contact between supports and the part for convex polyhedra, and show how to minimize the stair-step error for non-convex polyhedra. They also show how to reconcile multiple design

criteria simultaneously [23] and give support optimization algorithms for non-convex polygons [24]. (See also [22].) Schwerdt et al. show how to choose a build direction that protects prescribed facets from being damaged by supports [30] and to minimize the number of layers [31]. Recently, Agarwal and Desikan [1] have given an efficient algorithm to approximate a build direction which minimizes the contact-area for a convex polyhedron. Johnson [20] shows how to compute support descriptions efficiently for a given build direction and McMains and Séquin [27] show how to slice a model efficiently using the plane sweep paradigm. Barequet and Kaplan [6] describe the design of a software front-end for automating many process planning functions in LM.

There has also been substantial effort from the computer-aided design community, which is primarily experimental/heuristic in nature. Representative work here includes that of Frank and Fadel [17] who consider support optimization in the framework of an expert system, Bablani and Bagchi [4] who were the first to quantify the notion of stair-step error, Allen and Dutta [2] who give heuristics for minimizing support contact-area, Bøhn [8] who addresses the problem of automatic repair of computer models, and Kulkarni and Dutta [21] and Dolenc and Mäkelä [14] who consider the problem of slicing with variable layer thicknesses to capture fine detail in the model. The special issue by Stucki et al. [32], contains a wealth of information on commercial and university-based work in RP and LM.

1.2. A decomposition-based approach

All the process-planning algorithms for LM that we are aware of work by viewing the model as a single, monolithic unit. By contrast, the approach that we propose works by decomposing the model into a small number of pieces, building each piece separately using LM, and then gluing the built pieces together to obtain the physical prototype. (The number of pieces generated can be controlled by the user.) Specifically, given a decomposition direction, we decompose the model by intersecting it with a suitable plane perpendicular to this direction; we then build the pieces that lie in the same halfspace in the direction given by the normal to the plane that is contained in the halfspace. Fig. 2 illustrates this (in 2D, for convenience).

The decomposition-based approach has the advantage that it allows the construction of large models that cannot be accommodated in the workspace as a single piece. Moreover, the model can be built rapidly by building the pieces in parallel. Speed is particularly important in the "look and feel" prototyping that is the dominant use of RP today, since the goal here is to quickly get into the hands of the designer a physical version of the model to assess its general feel and appearance, and to rapidly iterate on the design until it is acceptable. In this context, the potential disadvantages of the decomposition-based approach (i.e., inaccuracies that may result from manually gluing the pieces together and reduced strength across the glued sections) are offset by the ability to build large models quickly in parallel.

A less obvious, but nevertheless crucial, advantage is that the support requirements of the decomposition-based approach are often much less than those of the conventional approach.[5] For instance, if a hollow sphere is built the conventional way, supports will be needed in the interior void and below the lower hemisphere; however, if it is built as two hemispheric shells, in opposite directions, then supports will be needed only in the regions previously occupied by the void, which results in an overall reduction in both support volume and contact-area. Our experiments confirm substantial savings for various other

[5] They can never be more, since the conventional approach corresponds to the special case where the decomposition plane coincides with the platform.

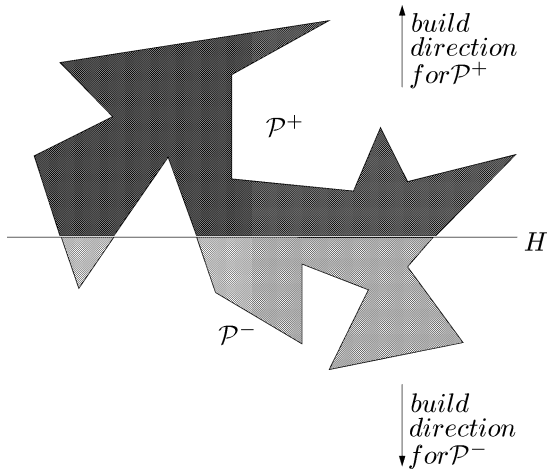

Fig. 2. The decomposition-based approach, shown in 2D for convenience. The polyhedron $\mathcal{P}$ is decomposed by a plane H into polyhedra $\mathcal{P}^+$ and $\mathcal{P}^-$, which are then built in the indicated directions.

models as well. Reducing support requirements is important because this translates into lower material costs and faster build times.

1.3. Contributions

We give the first provably correct and efficient geometric algorithms to decompose polyhedral (i.e., STL) models, with respect to a given decomposition direction, so that the support contact-area and, independently, the support volume is minimized. We formalize this problem in Section 2. In Sections 3 and 4, we consider convex polyhedra and devise efficient, plane-sweep-based algorithms. We show how to generate expressions for the support volume and contact-area as a function of the height of the sweep plane, and how to optimize them during the sweep. We discuss our implementation of these algorithms and the results of our experiments on convex polyhedra with up to 200,000 vertices.

It may be argued (quite reasonably) that real-life models are generally not convex, so the direct relevance of the above algorithms is somewhat limited. Our original motivation for considering convex polyhedra was to get a handle on the problem for the non-convex case, which is considerably more difficult because of the complex structure of the supports (see Fig. 3). Indeed, we will see that our approach for the convex case is one of the key building blocks to our solution for the non-convex case. In Section 5, we show how to handle non-convex polyhedra by first identifying certain critical facets (or parts thereof) using a technique called cylindrical decomposition [29]. It turns out that the problem can then be solved by applying to these critical facets the algorithm developed for the convex case. We also give experimental results for typical non-convex polyhedra. Another key problem encountered in the non-convex case (but not in the convex case) is to keep small the number of pieces in the decomposition. We give an efficient technique for controlling the size of the decomposition of non-convex polyhedra, so that the number of pieces generated is within a user-specified limit.

To the best of our knowledge, the only prior work related to this approach is due to Fekete and Mitchell [16]. They consider the problem of decomposing a polyhedron into special polyhedra called histograms, which can be built on certain “base” facets with no supports. They prove that deciding if a

polyhedron of genus zero can be decomposed into k histograms is NP-complete (this is true even in two dimensions for a polygon with holes). Our work differs from [16] in that we seek a decomposition (with respect to a given direction) into two polyhedra for which the total support requirement is minimized, but not necessarily zero.

2. Formalizing the problem

We denote by $\mathcal{P}$ the polyhedron of interest. We assume the facets of $\mathcal{P}$ are triangles and that its boundary is represented in some standard form, such as, for instance, a doubly-connected edge list [13] or a winged-edge structure [7]. (If necessary, such a representation can be computed easily from the standard STL representation of $\mathcal{P}$ [27].) Let $\boldsymbol{d}$ be a given *decomposition direction* (a unit-vector); we assume, without lost of generality, that $\boldsymbol{d}$ coincides with the positive z direction. Let H be any plane perpendicular to $\boldsymbol{d}$ and intersecting $\mathcal{P}$; we call H the *decomposition plane*. Let $\mathcal{P}^+$ be the closed polyhedron bounded by the facets of $\mathcal{P}$ (or portions thereof) that are above H, and by the facet $\mathcal{P} \cap H$. (Note that $\mathcal{P}^+$ may consist of more than one connected component.) If no part of $\mathcal{P}$ lies strictly above H, then $\mathcal{P}^+$ is taken to be empty. Define $\mathcal{P}^-$ symmetrically with respect to the portion of $\mathcal{P}$ that is below H. (Facet $\mathcal{P} \cap H$ appears in both $\mathcal{P}^+$ and $\mathcal{P}^-$, with outward normals $-\boldsymbol{d}$ and $\boldsymbol{d}$, respectively.) We define the *build direction* for $\mathcal{P}^+$ to be $\boldsymbol{d}$ and for $\mathcal{P}^-$ to be $-\boldsymbol{d}$, and we take H to be the *platform* for both polyhedra.

Let f be any facet of $\mathcal{P}$. We classify f, with respect to the given decomposition direction $\boldsymbol{d}$, as a *front facet*, a *back facet*, or a *parallel facet* of $\mathcal{P}$ depending on whether the angle between the decomposition direction $\boldsymbol{d}$ and the outward unit-normal, $\boldsymbol{n}_f$, of f is less than, greater than, or equal to 90°, respectively. (Note that f may be completely within $\mathcal{P}^+$ or $\mathcal{P}^-$, or may be partially in $\mathcal{P}^+$ and partially in $\mathcal{P}^-$. However, since the classification is done with respect to the fixed decomposition direction, rather than the build directions for $\mathcal{P}^+$ and $\mathcal{P}^-$, f has the same classification in $\mathcal{P}^+$ and/or $\mathcal{P}^-$ as it does in $\mathcal{P}$.)

We now formalize the notion of supports. A facet of a polyhedron will need to be supported iff the angle between its outer normal and the build direction of the polyhedron is greater than 90°. This implies that the back facets of $\mathcal{P}^+$ and the front facets of $\mathcal{P}^-$ will need to be supported. For concreteness, consider a back facet f of $\mathcal{P}^+$. The *support polyhedron* for f is the closure of the set of all points $p \in \mathbb{R}^3$ such that p is not in the interior of $\mathcal{P}^+$ and the ray shot from p in direction $\boldsymbol{d}$ first enters $\mathcal{P}^+$ through f. Informally, the support polyhedron of f is bounded from above by f, on the sides by vertical facets that

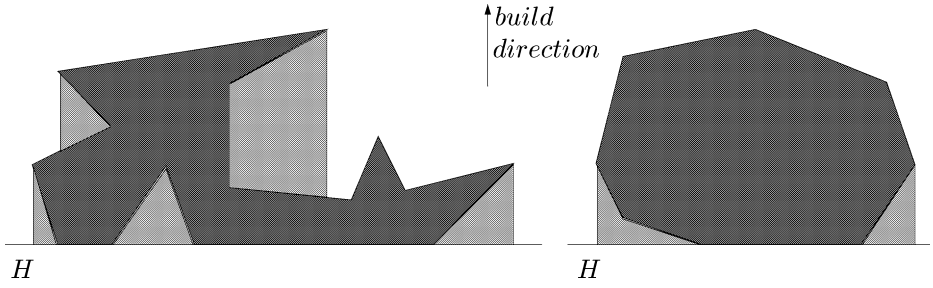

Fig. 3. Support structures (in light shading), shown in 2D for convenience. Supports in the non-convex case (left) exhibit complexities not seen in the convex case (right): (i) they can rest partially on other parts of the polyhedron; (ii) only a fraction of a facet may be in contact with supports; (iii) parallel facets can also be in contact with supports.

"drop down" from the edges of f, and from below by the platform and/or portions of front facets of $\mathcal{P}^+$. (If $\mathcal{P}^+$ is convex, then it is bounded from below by only the platform.) See Fig. 3 for an example in 2D.

The *support contact-area* for $\mathcal{P}^+$ is the total surface area of $\mathcal{P}^+$ that is in contact with supports. It includes the area of all the back facets of $\mathcal{P}^+$, except $\mathcal{P} \cap H$, and the areas of those portions of front facets and parallel facets that are in contact with supports. (Facet $\mathcal{P} \cap H$ rests on the platform and hence needs no supports. Note that while back facets are completely in contact with supports, front and parallel facets may be only partially in contact.) The *support volume* for $\mathcal{P}^+$ is the total volume of the support polyhedra for all back facets f of $\mathcal{P}^+$ (excluding $\mathcal{P} \cap H$). A symmetric discussion applies to the polyhedron $\mathcal{P}^-$.

We are now ready to state formally the problem that we wish to solve.

Problem 2.1. Given a polyhedron $\mathcal{P}$, with n vertices, and a decomposition direction $\boldsymbol{d}$, compute a plane H perpendicular to $\boldsymbol{d}$ which decomposes $\mathcal{P}$ into polyhedra $\mathcal{P}^+$ and $\mathcal{P}^-$ (as defined above) such that the total support contact-area or support volume is minimized when $\mathcal{P}^+$ and $\mathcal{P}^-$ are built in directions $\boldsymbol{d}$ and $-\boldsymbol{d}$, respectively. Additionally, if the user specifies an integer K, then the plane H should be optimal over all planes that generate no more than a total of K connected components of $\mathcal{P}^+$ and $\mathcal{P}^-$.

3. Decomposing a convex polyhedron to minimize contact-area of supports

In this section and in Section 4, we assume that $\mathcal{P}$ is convex. Thus $\mathcal{P}^+$ and $\mathcal{P}^-$ are both convex and the support polyhedron for a back facet of $\mathcal{P}^+$, or a front facet of $\mathcal{P}^-$, extends from the facet all the way to the platform, without intersecting any other facet (Fig. 3). Furthermore, parallel facets will not be in contact with supports.[6] These properties lead to an efficient algorithm for Problem 2.1, which we then incorporate into a solution for non-convex polyhedra.

Our approach is based on sweeping the plane H upwards starting from the xy-plane (we assume without lost of generality, that $\mathcal{P}$ lies above the xy-plane). Let H currently be at height h above the xy-plane and let f be any facet of $\mathcal{P}$. We classify f as an active or an inactive facet with respect to H depending on whether or not f is cut by H. Intuitively, an inactive facet is completely contained in $\mathcal{P}^+$ or $\mathcal{P}^-$ and its contribution to the contact-area is not affected by small "local" movements of H. On the other hand, an active facet is contained partially in $\mathcal{P}^+$ and partially in $\mathcal{P}^-$, and small movements of H affect the facet's contribution to the contact-area. Formally, we call f an *active facet with respect to* H if $H \cap f \neq \emptyset$ and at least one vertex of f is strictly above H. Otherwise, we call f an *inactive facet with respect to* H.

It follows that if f is an inactive front facet then its contribution to the total contact-area is $\mathit{area}(f)$ if it is in $\mathcal{P}^-$, and zero if it is in $\mathcal{P}^+$, where $\mathit{area}(f)$ is the area of f. If f is an active front facet, then only the part, f^-, of f that lies below H is in $\mathcal{P}^-$, so f contributes $\mathit{area}(f^-)$ to the total contact-area. Symmetrically, if f is an inactive back facet, then its contribution to the total contact-area is $\mathit{area}(f)$ if it is in $\mathcal{P}^+$, and zero if it is in $\mathcal{P}^-$. If f is an active back facet, then only the part f^+ of f that lies above H is in $\mathcal{P}^+$, so f contributes $\mathit{area}(f^+)$ to the total contact-area.

The expression for the total contact-area of $\mathcal{P}^+$ and $\mathcal{P}^-$ consists of two terms: the *inactive-area* term, which is the contact-area contributed by the inactive facets, and the *active-area* term, which is the contact-area contributed by the active facets. If we move H up or down, without crossing a vertex, then the

[6] Indeed, all references to facets in this section and in Section 4 mean non-parallel facets.

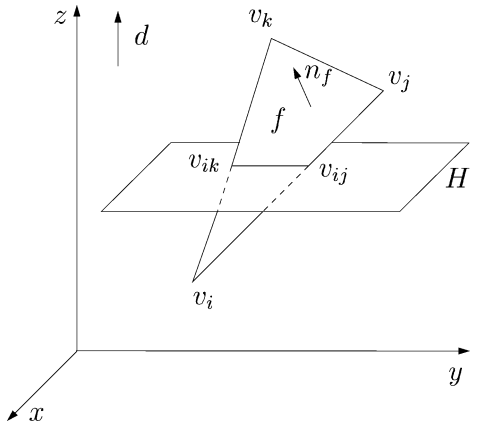

Fig. 4. Intersection of H and active front facet f.

inactive-area does not change, since inactive facets become active, and vice versa, only when H crosses a vertex. Therefore, the inactive-area term is simply a real number. However, the active-area changes because the fraction of an active facet that contributes to the total contact-area changes as H is moved. Lemma 3.1 below shows that the active-area term is a quadratic expression in h.

Lemma 3.1. *Let the plane H be at height h above the xy-plane. The total contact-area contributed by the active facets* (*i.e., the active-area*) *is of the form $Ah^2 + Bh + C$, where the coefficients A, B, and C depend only on the coordinates of the vertices of the active facets.*

Proof. Let f be any active facet, with vertices $v_i = (x_i, y_i, z_i)$, $v_j = (x_j, y_j, z_j)$, and $v_k = (x_k, y_k, z_k)$ (Fig. 4). We will prove that the contact-area contributed by f is of the form $a_f h^2 + b_f h + c_f$, where the coefficients a_f, b_f, and c_f depend only on the coordinates of the vertices of f. This implies the result.

Note that by definition of an active facet, f is not contained in H. Without lost of generality, assume that f is a front facet, that $z_i < z_j \leqslant z_k$, and that H intersects f between v_i and v_j. (The other cases are similar and are discussed at the end of the proof.) Let H intersect edge $\overline{v_i v_j}$ at $v_{ij} = (x_{ij}, y_{ij}, z_{ij})$ and edge $\overline{v_i v_k}$ at $v_{ik} = (x_{ik}, y_{ik}, z_{ik})$. We compute these intersection points as follows.

Let ℓ_{ij} and ℓ_{ik} be the lines containing $\overline{v_i v_j}$ and $\overline{v_i v_k}$, respectively. For real-valued parameters t and r, the equations of these two lines are:

$$\ell_{ij}: \quad x = x_i + (x_j - x_i)t, \quad y = y_i + (y_j - y_i)t, \quad z = z_i + (z_j - z_i)t,$$

and

$$\ell_{ik}: \quad x = x_i + (x_k - x_i)r, \quad y = y_i + (y_k - y_i)r, \quad z = z_i + (z_k - z_i)r.$$

Since $z_{ij} = z_{ik} = h$, the parameters t and r are given by

$$t = \frac{h - z_i}{z_j - z_i}, \qquad r = \frac{h - z_i}{z_k - z_i}.$$

Let $\alpha_j = (x_j - x_i)/(z_j - z_i)$, $\alpha_k = (x_k - x_i)/(z_k - z_i)$, $\beta_j = (y_j - y_i)/(z_j - z_i)$, and $\beta_k = (y_k - y_i)/(z_k - z_i)$. Then it follows that

$$x_{ij} = x_i + \alpha_j(h - z_i), \qquad y_{ij} = y_i + \beta_j(h - z_i)$$

and

$$x_{ik} = x_i + \alpha_k(h - z_i), \qquad y_{ik} = y_i + \beta_k(h - z_i).$$

Let f^- be the part of f below H; f^- is a triangle with vertices v_i, v_{ij}, and v_{ik}. Then

$$2 \cdot area(f^-) = \left|(\boldsymbol{v}_{ij} - \boldsymbol{v}_i) \times (\boldsymbol{v}_{ik} - \boldsymbol{v}_i)\right|,$$

where $\boldsymbol{v}_{ij} - \boldsymbol{v}_i = \alpha_j(h - z_i)\boldsymbol{i} + \beta_j(h - z_i)\boldsymbol{j} + (h - z_i)\boldsymbol{k}$ and $\boldsymbol{v}_{ik} - \boldsymbol{v}_i = \alpha_k(h - z_i)\boldsymbol{i} + \beta_k(h - z_i)\boldsymbol{j} + (h - z_i)\boldsymbol{k}$, and $|\cdot|$ denotes the length of the associated vector. Therefore,

$$\begin{aligned}(\boldsymbol{v}_{ij} - \boldsymbol{v}_i) \times (\boldsymbol{v}_{ik} - \boldsymbol{v}_i) &= \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ \alpha_j(h - z_i) & \beta_j(h - z_i) & (h - z_i) \\ \alpha_k(h - z_i) & \beta_k(h - z_i) & (h - z_i) \end{vmatrix} \\ &= (h - z_i)^2(\beta_j - \beta_k)\boldsymbol{i} - (h - z_i)^2(\alpha_j - \alpha_k)\boldsymbol{j} + (h - z_i)^2(\alpha_j\beta_k - \alpha_k\beta_j)\boldsymbol{k}.\end{aligned}$$

It follows that

$$area(f^-) = \frac{1}{2}(h - z_i)^2\left((\beta_j - \beta_k)^2 + (\alpha_j - \alpha_k)^2 + (\alpha_j\beta_k - \alpha_k\beta_j)^2\right)^{1/2}.$$

The coefficient of $(h - z_i)^2$ above is a constant which depends only on the coordinates of the vertices of f. In fact, it is easy to verify that this constant is equal to $area(f)/((z_j - z_i)(z_k - z_i))$. Hence the contribution of f to the active-area term is of the form $a_f h^2 + b_f h + c_f$, as claimed.

If $z_i = z_j$ or if H intersects f between v_j and v_k, then f^- is a quadrilateral and $area(f^-) = area(f) - area(f^+)$, where f^+ is the part of f above H. Since f^+ is a triangle, its area can be written as a quadratic expression similar to the one above. Moreover, $area(f)$ is a constant. It follows that $area(f^-)$ is of the claimed form. Finally, if f is a back facet, then a symmetric discussion applies.

Summing the area contributions of all active facets gives the lemma. □

Remark 3.1. Note that the coefficient A in Lemma 3.1 is the sum of coefficients a_f for those active facets f that contribute triangular areas to the total contact-area minus the sum of coefficients a_g for those active facets g that contribute quadrilateral areas to the total contact-area. Therefore, it is possible that $A = 0$. Similarly, it is also possible that $B = 0$. In fact, the following example shows that A and B can be zero simultaneously.

Let $\mathcal{P}$ be a cube which has been sheared by a small amount in the positive y direction. That is, if we view $\mathcal{P}$ from the positive x-axis, then the top and bottom facets are parallel to the xy-plane, the front and back facets are parallel to the yz-plane, and the left and right facets are parallel but slanted rightwards. Let F and G be the left and right facets, respectively. Split F into triangles f and f' and G into triangles g and g'. Let f be the triangle of F with one vertex at the bottom and two at the top, and let g be the similar triangle of G. Note that $area(f) = area(g)$, and, with respect to the positive z decomposition direction, f is a front facet and g is a back facet. Let z_i, z_j, and z_k $(= z_j)$ be the z-coordinates of the bottom, middle, and top vertices of f (and g).

Let H be anywhere between z_i and z_j. Then f's contribution to the contact-area is a triangle whose area is $(h - z_i)^2 area(f)/((z_j - z_i)(z_k - z_i))$, and g's contribution is a quadrilateral whose area is $area(g) - (h - z_i)^2 area(g)/((z_j - z_i)(z_k - z_i))$. Therefore, the total contribution of f and g is simply $area(g)$. A symmetric discussion shows that the total contribution of f' and g' is $area(f')$. It follows that the total contact-area of $\mathcal{P}$ for this position of H is a constant equal to $area(g) + area(f')$, i.e., $A = B = 0$.

In the algorithm to follow, we will need to optimize the active-area formula over a certain range of h for which it is valid. We will show there how to handle these special cases.

3.1. The algorithm

We sweep the plane H upwards, starting at the lowest vertex of $\mathcal{P}$. We stop at each vertex and update the inactive-area and the active-area terms, based on the facets incident to the current vertex. We then minimize the active-area term (hence the total contact-area) as the position, h, of H ranges between the z-coordinates of the current vertex and the next one. This gives the optimal position for H between successive vertices. We repeat this for all vertices to find the globally-best position for H.

Here is the algorithm in more detail:

- In a preprocessing step, we sort the vertices of $\mathcal{P}$ according to non-decreasing z-coordinates, as v_1, $v_2, \ldots, v_n$ (ties are broken arbitrarily). For each facet $f \in \mathcal{P}$, we determine whether it is a front, a back, or a parallel facet, and also compute *area*(f). We set the active-area term identically equal to zero and the inactive-area term equal to the total area of the back facets. We set the current minimum contact-area equal to the sum of these two terms. We then scan the vertices in their sorted order.
- Let v_ℓ be the current vertex, $1 \leqslant \ell < n$. For each facet f incident to v_ℓ, we do the following:

  Case 1: v_ℓ *is the lowest vertex of* f. (Thus, f changes from inactive to active at v_ℓ.) If f is a back facet, then we subtract *area*(f) from the inactive-area term. Using Lemma 3.1, we compute and add the expression $a_f h^2 + b_f h + c_f$ to the active-area term, thereby including the contribution of f^+ to the total contact-area when H is between z_ℓ and $z_{\ell+1}$. If f is a front facet, then we update only the active-area term to include the contribution of f^- to the total contact-area.

  Case 2: v_ℓ *is the highest vertex of* f. (Thus, f changes from active to inactive at v_ℓ.) This case is symmetric to Case 1. That is, if f is a back facet then we use Lemma 3.1 to compute and subtract the expression $a_f h^2 + b_f h + c_f$ from the active-area term, thereby removing the contribution of f^+ to the total contact-area when H is between z_ℓ and $z_{\ell+1}$. If f is a front facet, then we update the active area term to remove the contribution of f^-, and we add *area*(f) to the inactive-area term.

  Case 3: v_ℓ *is the middle vertex of* f. Here f continues to be active, but the active-area term must be updated since H intersects a different edge of f above v_ℓ than it did below. We perform this update using Lemma 3.1.

  After one of the three cases above is executed for all facets f incident to v_ℓ, we have a new active-area term $Ah^2 + Bh + C$, which is valid for h in the interval $[z_\ell, z_{\ell+1}]$. We can minimize this term using standard techniques from calculus: If $A \neq 0$, then the minimum is attained at $h = -B/2A$ if $-B/2A \in [z_\ell, z_{\ell+1}]$, and at $h = z_\ell$ or at $h = z_{\ell+1}$ otherwise. If $A = 0$ and $B \neq 0$, then the minimum is attained at either $h = z_\ell$ or at $h = z_{\ell+1}$. If $A = B = 0$, then the active-area is constant in $[z_\ell, z_{\ell+1}]$, so we (arbitrarily) pick z_ℓ as the "optimum" value of h. In any case, once the optimum h has been determined, we compute the total contact-area for this h and update the current minimum contact-area, if necessary.

Theorem 3.1. *The contact-area version of Problem* 2.1 *can be solved in* $O(n \log n)$ *time for a convex polyhedron* $\mathcal{P}$ *with* n *vertices* ($O(n)$ *time if the vertices are given in sorted order in the decomposition direction* $\boldsymbol{d}$).

Proof. The correctness of the algorithm follows from the earlier discussion. The initialization time is dominated by the $O(n \log n)$ time to sort the vertices. At each vertex v_ℓ, the incident faces can be accessed in constant time apiece, since $\mathcal{P}$ is given as a doubly-connected edge list. The time to update the active-area and inactive-area terms is proportional to the degree of v_ℓ. The minimization of the active-area term takes constant time. Summing over all vertices, the sweep time is $O(n)$ since the sum of the vertex degrees in $\mathcal{P}$ is $O(n)$. The claimed time bound follows. □

3.2. Experimental results

We have implemented the above algorithm and tested it on several convex polyhedra. The program is written in C++, and performs floating point computations in double-precision. The tests were done on a SUN Ultra 10 Sparc machine with 256 MB of main memory and a 440 MHz processor.

We generated two classes of test polyhedra. For the first class, we generated for each n, a set of n points at random on a sphere of radius 100 centered at the origin. (In our experiments, n ranged from 20,000 to 200,000, in steps of 20,000.) We then computed the convex hull of these n points using the `qhull` program [10], and used it as our test polyhedron. (Since the n points were in convex position, all of them appeared on the convex hull. Also, we set the parameters to `qhull` such that it did not merge coplanar facets; thus all facets were triangles.) Table 1 shows the running times of our program and the minimum contact-area computed. To illustrate the savings realized by decomposition, we also computed the contact-area without decomposition in each case.

Table 1
Minimum support contact-area for convex polyhedra generated from random points on a sphere of radius 100. Here "non-decomp. contact-area" refers to the support contact-area when the polyhedra are built without decomposition; observe the significant reductions achieved via decomposition. Note that the support contact-area without decomposition is roughly equal to the area of the lower hemisphere, as it should be

Support contact-area				
#verts n	min contact-area	$h_{\min}$	run time (s)	non-decomp. contact-area
20,000	579.0	−0.14	0.6	62,708.1
40,000	424.2	0.04	1.2	62,832.6
60,000	338.6	−0.04	2.0	62,803.7
80,000	286.6	−0.05	2.7	62,802.5
100,000	274.6	0.02	3.5	62,840.0
120,000	243.6	0.05	4.2	62,864.0
140,000	226.4	0	5.0	62,824.7
160,000	211.6	−0.01	5.8	62,833.2
180,000	191.2	0.03	6.5	62,845.1
200,000	174.7	−0.02	7.4	62,814.9

Table 2
Minimum support contact-area for convex polyhedra generated from random points on a cone; each polyhedron has been rotated by the indicated angle to make it non-symmetric about the origin. Here "non-decomp. contact-area" refers to the support contact-area when the polyhedra are built without decomposition; observe the significant reductions achieved via decomposition

		Support contact-area			
#verts n	angle	min contact-area	$h_{\min}$	mean run time (s)	non-decomp. contact-area
20,001	200	7,752.6	−3.2	0.6	55,300.5
	37	11,705.0	2.3		57,111.7
40,001	40	12,193.1	2.7	1.3	58,304.3
	75	3,973.7	0.7		52,005.6
60,001	240	7,037.8	−0.5	2.0	55,082.1
	112	5,315.5	−0.5		55,068.2
80,001	80	2,714.5	0	2.8	52,272.1
	150	10,108.5	−1.5		51,825.6
10,0001	280	2,649.1	0	3.6	52,160.8
	187	3,447.8	−3.6		59,820.0
120,001	120	7,018.4	−0.2	4.3	55,068.5
	225	11,908.1	−1.0		49,168.8
140,001	320	12,209.7	1.2	5.1	58,237.1
	263	1,967.0	0.1		54,011.3
160,001	160	7,657.2	−2.1	5.9	55,325.7
	300	6,962.6	0.2		51,425.1
180,001	0	5.0	1.7	6.7	44,798.2
	338	8,175.7	1.5		51,938.3
200,001	200	7,625.8	−2.1	7.5	55,297.5
	15	6,198.2	3.4		49,509.5

As one might expect, the test polyhedra generated above were quite symmetric about the origin. As a result, the optimum decomposition plane was always close to the xy-plane. Therefore, we also generated a second set of polyhedra that did not exhibit such symmetry. For this, we used `qhull` to generate $n+1$ points at random on a cone whose major axis was along the z-axis, for n in the range 20,000–200,000 (the additional point was the apex of the cone). We then rotated the cone by a randomly chosen angle to

make it non-symmetric about the origin. For each n, we generated two non-symmetric test polyhedra in this fashion. Table 2 shows our results; here "angle" refers to the angle by which the cone was rotated. For each input size, the running times on the two polyhedra were nearly the same; therefore, we averaged the times. This closeness in run times is to be expected since the complexity of the algorithm depends primarily on the graph structure of $\mathcal{P}$, which is the same regardless of orientation. In fact, since the graph is fully triangulated, the number of edges and faces are the same for different polyhedra with the same number of vertices. This is borne out by Tables 1 and 2.

3.3. Discussion

One might wonder if there is always an optimal solution where the plane H passes through a vertex of $\mathcal{P}$, thus obviating the need to minimize the active-area formula in between successive vertices. In this section, we show that this is not the case by proving that for the pyramid $\mathcal{P}$ of Fig. 5 the optimal plane does not contain any vertex.

In what follows, we will use the notation $\langle v_i, v_j, v_k\rangle$ to denote a facet (triangle) of $\mathcal{P}$ that is bounded by vertices v_i, v_j, and v_k.

It is easy to verify that

$$2 \cdot \mathit{area}\big(\langle v_1, v_3, v_2\rangle\big) = \big|(\boldsymbol{v}_3 - \boldsymbol{v}_1) \times (\boldsymbol{v}_2 - \boldsymbol{v}_1)\big| = \sqrt{6},$$

$$2 \cdot \mathit{area}\big(\langle v_1, v_2, v_4\rangle\big) = \big|(\boldsymbol{v}_4 - \boldsymbol{v}_1) \times (\boldsymbol{v}_2 - \boldsymbol{v}_1)\big| = \sqrt{6},$$

$$2 \cdot \mathit{area}\big(\langle v_2, v_3, v_4\rangle\big) = \big|(\boldsymbol{v}_3 - \boldsymbol{v}_2) \times (\boldsymbol{v}_4 - \boldsymbol{v}_2)\big| = \sqrt{8}.$$

Note that facet $\langle v_1, v_3, v_4\rangle$ is a parallel facet for the indicated decomposition direction $\boldsymbol{d}$, and, therefore, does not contribute to the contact-area.

We now compute the total contact-area for $\mathcal{P}^+$ and $\mathcal{P}^-$ for each decomposition plane H that passes through a vertex of $\mathcal{P}$.

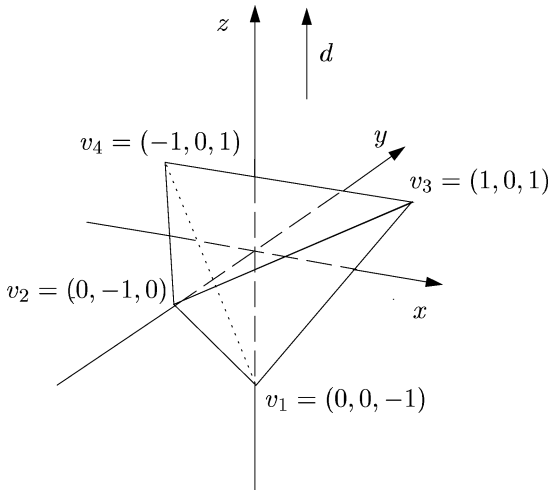

Fig. 5. A polyhedron for which the decomposition plane realizing the minimum contact-area does not pass through a vertex.

Case 1. *H passes through v_1, i.e., H is the plane $z = -1$.*
In this case, $\mathcal{P}^-$ is empty and the (active) back facets $\langle v_1, v_3, v_2\rangle$ and $\langle v_1, v_2, v_4\rangle$ contribute $area(\langle v_1, v_3, v_2\rangle)$ and $area(\langle v_1, v_2, v_4\rangle)$, respectively, to the contact-area for $\mathcal{P}^+$. Therefore, the total contact-area is $\sqrt{6}/2 + \sqrt{6}/2 = 2.449$.

Case 2. *H passes through v_2, i.e., H is the plane $z = 0$.*
Let H intersect edges $\overline{v_1v_3}$ and $\overline{v_1v_4}$ at points v_{13} and v_{14}, respectively. We have $v_{13} = (0.5, 0, 0)$ and $v_{14} = (-0.5, 0, 0)$. There are no facets that contribute to the contact-area for $\mathcal{P}^-$. The (active) back facets $\langle v_1, v_3, v_2\rangle$ and $\langle v_1, v_2, v_4\rangle$ contribute $area(\langle v_2, v_{13}, v_3\rangle)$ and $area(\langle v_2, v_4, v_{14}\rangle)$, respectively, to the contact-area for $\mathcal{P}^+$. We have

$$2 \cdot area\big(\langle v_2, v_{13}, v_3\rangle\big) = \big|(\boldsymbol{v}_{13} - \boldsymbol{v}_2) \times (\boldsymbol{v}_3 - \boldsymbol{v}_2)\big| = \sqrt{1.5},$$

$$2 \cdot area\big(\langle v_2, v_4, v_{14}\rangle\big) = \big|(\boldsymbol{v}_{14} - \boldsymbol{v}_2) \times (\boldsymbol{v}_4 - \boldsymbol{v}_2)\big| = \sqrt{1.5}.$$

Therefore, the total contact-area is $\sqrt{1.5}/2 + \sqrt{1.5}/2 = 1.225$.

Case 3. *H passes through v_3 (and v_4), i.e., H is the plane $z = 1$.*
Here $\mathcal{P}^+$ is empty. Only the (inactive) front facet $\langle v_2, v_3, v_4\rangle$ contributes to the contact-area for $\mathcal{P}^-$. Its contribution is $area(\langle v_2, v_3, v_4\rangle) = \sqrt{2} = 1.414$, which is also the total contact-area.

Now, let H be the plane $z = 0.5$. This plane does not pass through any vertex of $\mathcal{P}$. We will show that the total contact-area for this plane is smaller than any of the contact-areas computed in Cases 1–3 above.

Let H intersect edges $\overline{v_1v_3}$, $\overline{v_1v_4}$, $\overline{v_2v_3}$, and $\overline{v_2v_4}$ at points v_{13}, v_{14}, v_{23}, and v_{24}, respectively. We have $v_{13} = (0.75, 0, 0.5)$, $v_{14} = (-0.75, 0, 0.5)$, $v_{23} = (0.5, -0.5, 0.5)$, and $v_{24} = (-0.5, -0.5, 0.5)$.

Facet $\langle v_2, v_3, v_4\rangle$ is a (active) front facet and contributes $area(\langle v_2, v_{23}, v_{24}\rangle)$ to the contact-area for $\mathcal{P}^-$. Facets $\langle v_1, v_3, v_2\rangle$ and $\langle v_1, v_2, v_4\rangle$ are (active) back facets, and contribute $area(\langle v_{23}, v_{13}, v_3\rangle)$ and $area(\langle v_{14}, v_4, v_{24}\rangle)$, respectively, to the contact-area for $\mathcal{P}^+$. We have

$$2 \cdot area\big(\langle v_2, v_{23}, v_{24}\rangle\big) = \big|(\boldsymbol{v}_{23} - \boldsymbol{v}_2) \times (\boldsymbol{v}_{24} - \boldsymbol{v}_2)\big| = \frac{1}{\sqrt{2}},$$

$$2 \cdot area\big(\langle v_{23}, v_{13}, v_3\rangle\big) = \big|(\boldsymbol{v}_{23} - \boldsymbol{v}_3) \times (\boldsymbol{v}_{13} - \boldsymbol{v}_3)\big| = \frac{\sqrt{1.5}}{4},$$

$$2 \cdot area\big(\langle v_{14}, v_4, v_{24}\rangle\big) = \big|(\boldsymbol{v}_{14} - \boldsymbol{v}_4) \times (\boldsymbol{v}_{24} - \boldsymbol{v}_4)\big| = \frac{\sqrt{1.5}}{4}.$$

Thus, the total contact-area is $(1/2)1/\sqrt{2} + (1/2)\sqrt{1.5}/4 + (1/2)\sqrt{1.5}/4 = 0.660$, which is less than any of the contact-areas computed in Cases 1–3. This proves that the decomposition plane that realizes the minimum contact-area need not pass through any vertex of $\mathcal{P}$. It turns out that, for this example, the optimal plane is $H\colon\ z = 0.46$, and the corresponding total contact-area is 0.656. (This was computed by our program in Section 3.2.)

4. Decomposing a convex polyhedron to minimize volume of supports

In this section, we describe an algorithm to decompose $\mathcal{P}$ such that when the resulting polyhedra $\mathcal{P}^+$ and $\mathcal{P}^-$ are built in directions $\boldsymbol{d}$ and $-\boldsymbol{d}$, respectively, the total volume of supports is minimized.

Each support polyhedron for $\mathcal{P}^+$ is bounded from above by a back facet, on the sides by vertical edges emanating from the edges of the facet, and from below by the decomposition plane H. A symmetric discussion applies for $\mathcal{P}^-$.

For any given decomposition plane H: $z = h$, the expression for the support volume consists of two terms. The *inactive-volume* term and the *active-volume* term, which are, respectively, the total volumes of the support polyhedra contributed by the inactive and the active facets. If we move H up or down without crossing a vertex, the active-volume as well as the inactive-volume changes (unlike contact-area, where only the active-area changes). Lemma 4.1 below proves that the active-volume term is cubic in h, and Lemma 4.2 shows that the inactive-volume is linear in h.

Lemma 4.1. *Let the plane H be at height h above the xy-plane. The total support volume contributed by the active facets* (*i.e., the active-volume*) *is of the form $Ah^3 + Bh^2 + Ch + D$, where the coefficients A, B, C, and D depend only on the coordinates of the vertices of the active facets.*

Proof. Let f be any active facet, with vertices $v_i = (x_i, y_i, z_i)$, $v_j = (x_j, y_j, z_j)$, and $v_k = (x_k, y_k, z_k)$, where $z_i < z_j$, $z_i < z_k$ and let the vertices v_i, v_j, and v_k be in counterclockwise order when viewed from the outside of $\mathcal{P}$ (i.e., $(\boldsymbol{v}_j - \boldsymbol{v}_i) \times (\boldsymbol{v}_k - \boldsymbol{v}_i)$ has the same direction as the outward unit-normal, $\boldsymbol{n}_f$, to f). Clearly, it suffices to prove that the support volume contributed by f is of the form $a_f h^3 + b_f h^2 + c_f h + d_f$, where the coefficients a_f, b_f, c_f, and d_f depend only on the coordinates of the vertices of f.

Note that, by definition of an active facet, f is not contained in H. Assume that H intersects edges $\overline{v_i v_j}$ and $\overline{v_i v_k}$, so that f^- is a triangle and f^+ is a quadrilateral. (The other cases are similar and are discussed at the end of the proof.) If f is a front facet, then it contributes a support polyhedron only to $\mathcal{P}^-$, whereas if f is a back facet, then it contributes a support polyhedron only to $\mathcal{P}^+$. Let us denote this support polyhedron by R_f. Let v_i', v_j', and v_k' be the vertical projections of v_i, v_j, and v_k, respectively, on H. If f is a front facet, then R_f is defined by the vertices v_i, v_i', v_{ij}, and v_{ik} (Fig. 6); if f is a back facet, then R_f is defined by the vertices v_{ij}, v_{ik}, v_j', v_k', v_j, and v_k (Fig. 7). Our goal is to find a formula for the volume, $vol(R_f)$, of R_f in each case.

Let S_j, $1 \leqslant j \leqslant r$ be the facets of R_f, for some positive integer r. Let $\boldsymbol{N}_j$ be the outward unit-normal to S_j ("outward" with respect to R_f, not $\mathcal{P}$). Let Q_j be any point on S_j and let $\boldsymbol{Q}_j$ be the position vector of Q_j. From [18] we have

$$vol(R_f) = \frac{1}{3}\left|\sum_j (\boldsymbol{Q}_j \cdot \boldsymbol{N}_j) area(S_j)\right|. \tag{1}$$

We next consider the two possibilities for f separately.

Case 1. *f is a front facet of $\mathcal{P}$* (Fig. 6).
In this case R_f is bounded by the following facets:

$$\begin{aligned} S_1: &\quad \langle v_i, v_{ij}, v_i' \rangle \\ S_2 = f^-: &\quad \langle v_i, v_{ik}, v_{ij} \rangle \\ S_3: &\quad \langle v_i, v_i', v_{ik} \rangle \\ S_4: &\quad \langle v_i', v_{ij}, v_{ik} \rangle \end{aligned}$$

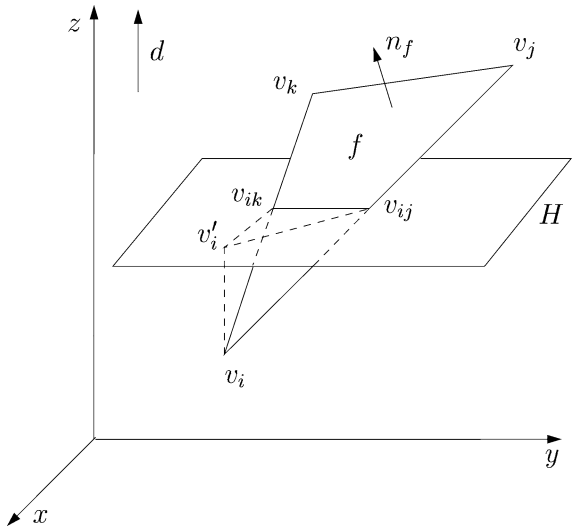

Fig. 6. The support polyhedron, R_f, contributed by an active front facet f. R_f is defined by the vertices v_i, v'_i, v_{ij}, and v_{ik}.

Table 3

Facet	Point Q_j	Normal $\boldsymbol{N}_j$
S_1	v_i	$(\boldsymbol{v}_j - \boldsymbol{v}_i) \times \boldsymbol{d}/\lvert(\boldsymbol{v}_j - \boldsymbol{v}_i) \times \boldsymbol{d}\rvert$
S_2	v_i	$-\boldsymbol{n}_f$
S_3	v_i	$\boldsymbol{d} \times (\boldsymbol{v}_k - \boldsymbol{v}_i)/\lvert\boldsymbol{d} \times (\boldsymbol{v}_k - \boldsymbol{v}_i)\rvert$
S_4	v'_i	$\boldsymbol{d}$

Table 3 shows $\boldsymbol{N}_j$ and the point Q_j for each facet S_j. We have chosen Q_j such that subsequent calculations become simple.

Note that all the dot products $\boldsymbol{Q}_j \cdot \boldsymbol{N}_j$, except for $\boldsymbol{Q}_4 \cdot \boldsymbol{N}_4$, depend only on f (and $\boldsymbol{d}$, which is fixed); the dot product $\boldsymbol{Q}_4 \cdot \boldsymbol{N}_4 = h$. We now generate an expression for each $area(S_j)$.

Expression for $area(S_1)$**:**

Consider the vertical projection, v''_i, of v_i on the plane passing through v_j and parallel to H.[7] Facet S_1 is simply the triangular portion of $\langle v_i, v_j, v''_i\rangle$ lying below H. From the proof of Lemma 3.1

$$area(S_1) = (h - z_i)^2 \cdot \frac{area(\langle v_i, v_j, v''_i\rangle)}{(z_j - z_i)(z_j - z_i)},$$

[7] To avoid clutter, we do not show v''_i in Fig. 6.

where $area(\langle v_i, v_j, v_i'' \rangle) = |(\boldsymbol{v}_j - \boldsymbol{v}_i) \times (\boldsymbol{v}_i'' - \boldsymbol{v}_i)|/2$, and $\boldsymbol{v}_i'' - \boldsymbol{v}_i$ has components $(0, 0, z_j - z_i)$. It follows that $area(S_1)$ can be written in the form

$$area(S_1) = a_f^{(1)} h^2 + b_f^{(1)} h + c_f^{(1)}, \tag{2}$$

for some coefficients $a_f^{(1)}$, $b_f^{(1)}$, and $c_f^{(1)}$ that depend only on the coordinates of the vertices of f.

Expression for $area(S_2)$**:**

From the proof of Lemma 3.1 it is immediate that

$$area(S_2) = a_f^{(2)} h^2 + b_f^{(2)} h + c_f^{(2)}, \tag{3}$$

for some coefficients $a_f^{(2)}$, $b_f^{(2)}$, and $c_f^{(2)}$ that depend only on the coordinates of the vertices of f.

Expression for $area(S_3)$**:**

The discussion here is identical to the one for $area(S_1)$, except that we use the vertical projection, v_i''', of v_i on the plane passing through v_k and parallel to H. Therefore,

$$area(S_3) = (h - z_i)^2 \cdot \frac{area(\langle v_i, v_k, v_i''' \rangle)}{(z_k - z_i)(z_k - z_i)},$$

where $area(\langle v_i, v_k, v_i''' \rangle) = |(\boldsymbol{v}_k - \boldsymbol{v}_i) \times (\boldsymbol{v}_i''' - \boldsymbol{v}_i)|/2$, and $\boldsymbol{v}_i''' - \boldsymbol{v}_i = (0, 0, z_k - z_i)$. It follows that $area(S_3)$ can be written in the form

$$area(S_3) = a_f^{(3)} h^2 + b_f^{(3)} h + c_f^{(3)}, \tag{4}$$

for some coefficients $a_f^{(3)}$, $b_f^{(3)}$, and $c_f^{(3)}$ that depend only on the coordinates of the vertices of f.

Expression for $area(S_4)$**:**

Now, $area(S_4)$ is simply the area of the projection of f^- on H, and is hence equal to $area(f^-)$ times the cosine of the angle between H and the plane containing f^-. Therefore, $area(S_4) = area(f^-)(\boldsymbol{n}_f \cdot \boldsymbol{d})$.

Recall from the proof of Lemma 3.1 that $area(f^-)$ is quadratic in h. It follows that $area(S_4)$ can be written in the form

$$area(S_4) = a_f^{(4)} h^2 + b_f^{(4)} h + c_f^{(4)}, \tag{5}$$

for some coefficients $a_f^{(4)}$, $b_f^{(4)}$, and $c_f^{(4)}$ that depend only on the coordinates of the vertices of f.

Let us denote by $active\text{-}volume_f(h)$ the support volume contributed by an active facet f with respect to the plane H: $z = h$. From Eqs. (2)–(5) and the fact that $\boldsymbol{Q}_4 \cdot \boldsymbol{N}_4 = h$, we have

$$\begin{aligned} active\text{-}volume_f(h) &= \frac{1}{3}\left|\sum_{j=1}^{4} (\boldsymbol{Q}_j \cdot \boldsymbol{N}_j)\left(a_f^{(j)} h^2 + b_f^{(j)} h + c_f^{(j)}\right)\right| \\ &= \frac{1}{3}\left|\sum_{j=1}^{3} (\boldsymbol{Q}_j \cdot \boldsymbol{N}_j)\left(a_f^{(j)} h^2 + b_f^{(j)} h + c_f^{(j)}\right) + h\left(a_f^{(4)} h^2 + b_f^{(4)} h + c_f^{(4)}\right)\right| \end{aligned}$$

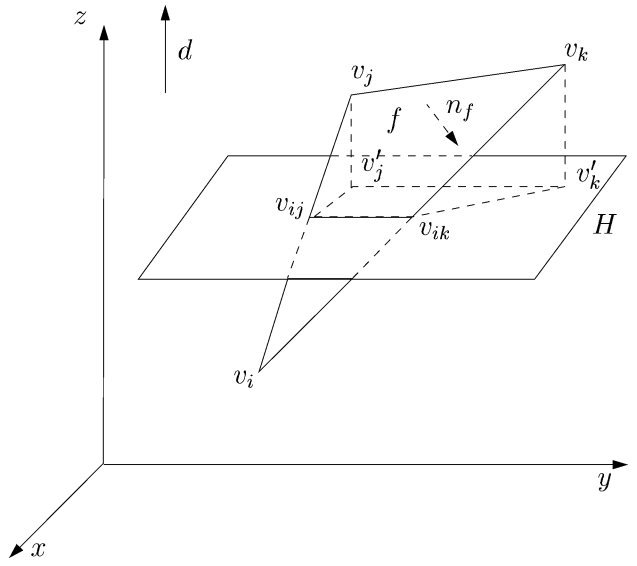

Fig. 7. The support polyhedron, R_f, contributed by an active back facet f. R_f is defined by the vertices v_j, v_k, v_{ij}, v_{ik}, v'_j, and v'_k.

$$= \frac{1}{3}\left| a_f^{(4)}h^3 + \left(b_f^{(4)} + \sum_{j=1}^{3}(\boldsymbol{Q}_j \cdot \boldsymbol{N}_j)a_f^{(j)}\right)h^2 + \left(c_f^{(4)} + \sum_{j=1}^{3}(\boldsymbol{Q}_j \cdot \boldsymbol{N}_j)b_f^{(j)}\right)h + \sum_{j=1}^{3}(\boldsymbol{Q}_j \cdot \boldsymbol{N}_j)c_f^{(j)}\right|.$$

Therefore,

$$\textit{active-volume}_f(h) = a_f h^3 + b_f h^2 + c_f h + d_f, \tag{6}$$

for some coefficients a_f, b_f, c_f, and d_f that depend only on the coordinates of the vertices of f.

Case 2. *f is a back facet of $\mathcal{P}$* (Fig. 7).
In this case R_f is bounded by the following facets:

$$\begin{aligned} S_1&: \quad \langle v_{ik}, v'_k, v_k\rangle \\ S_2 = f^+&: \quad \langle v_{ik}, v_k, v_j, v_{ij}\rangle \\ S_3&: \quad \langle v_{ij}, v_j, v'_j\rangle \\ S_4&: \quad \langle v_j, v_k, v'_k, v'_j\rangle \\ S_5&: \quad \langle v_{ij}, v'_j, v'_k, v_{ik}\rangle \end{aligned}$$

Table 4 shows $\boldsymbol{N}_j$ and the point Q_j for each facet S_j. Again, we have chosen Q_j such that subsequent calculations become simple.

Note that all the dot products $\boldsymbol{Q}_j \cdot \boldsymbol{N}_j$, except for $\boldsymbol{Q}_5 \cdot \boldsymbol{N}_5$, depend only on f (and $\boldsymbol{d}$, which is fixed); the dot product $\boldsymbol{Q}_5 \cdot \boldsymbol{N}_5 = -h$. We now generate an expression for each $\textit{area}(S_j)$.

Table 4

Facet	Point Q_j	Normal $\boldsymbol{N}_j$
S_1	v_k	$\boldsymbol{d} \times (\boldsymbol{v}_i - \boldsymbol{v}_k)/\lvert\boldsymbol{d} \times (\boldsymbol{v}_i - \boldsymbol{v}_k)\rvert$
S_2	v_k	$-\boldsymbol{n}_f$
S_3	v_j	$\boldsymbol{d} \times (\boldsymbol{v}_j - \boldsymbol{v}_i)/\lvert\boldsymbol{d} \times (\boldsymbol{v}_j - \boldsymbol{v}_i)\rvert$
S_4	v_j	$\boldsymbol{d} \times (\boldsymbol{v}_k - \boldsymbol{v}_j)/\lvert\boldsymbol{d} \times (\boldsymbol{v}_k - \boldsymbol{v}_j)\rvert$
S_5	v'_k	$-\boldsymbol{d}$

Expression for $area(S_1)$**:**

Consider the vertical projection, v''_k, of v_k on the plane passing through v_i and parallel to H.[8] Facet S_1 is simply the triangular portion of $\langle v_i, v_k, v''_k\rangle$ lying above H. From the proof of Lemma 3.1

$$area(S_1) = (h - z_k)^2 \cdot \frac{area(\langle v_i, v_k, v''_k\rangle)}{(z_i - z_k)(z_i - z_k)},$$

where $area(\langle v_i, v_k, v''_k\rangle) = |(\boldsymbol{v}_i - \boldsymbol{v}_k) \times (\boldsymbol{v}''_k - \boldsymbol{v}_k)|/2$, and $\boldsymbol{v}''_k - \boldsymbol{v}_k$ has components $(0, 0, z_i - z_k)$. It follows that $area(S_1)$ can be written in the form

$$area(S_1) = a_f^{(1)}h^2 + b_f^{(1)}h + c_f^{(1)}, \tag{7}$$

for some coefficients $a_f^{(1)}$, $b_f^{(1)}$, and $c_f^{(1)}$ that depend only on the coordinates of the vertices of f.[9]

Expression for $area(S_2)$**:**

From the proof of Lemma 3.1 it is immediate that

$$area(S_2) = a_f^{(2)}h^2 + b_f^{(2)}h + c_f^{(2)}, \tag{8}$$

for some coefficients $a_f^{(2)}$, $b_f^{(2)}$, and $c_f^{(2)}$ that depend only on the coordinates of the vertices of f.

Expression for $area(S_3)$**:**

The discussion is identical to the one above for $area(S_1)$, except that we use the projection v''_j of v_j on the plane passing through v_i and parallel to H. Therefore,

$$area(S_3) = (h - z_j)^2 \cdot \frac{area(\langle v_i, v_j, v''_j\rangle)}{(z_i - z_j)(z_i - z_j)},$$

where $area(\langle v_i, v_j, v''_j\rangle) = |(\boldsymbol{v}_i - \boldsymbol{v}_j) \times (\boldsymbol{v}''_j - \boldsymbol{v}_j)|/2$, and $\boldsymbol{v}''_j - \boldsymbol{v}_j$ has components $(0, 0, z_i - z_j)$. It follows that $area(S_3)$ can be written in the form

$$area(S_3) = a_f^{(3)}h^2 + b_f^{(3)}h + c_f^{(3)}, \tag{9}$$

for some coefficients $a_f^{(3)}$, $b_f^{(3)}$, and $c_f^{(3)}$ that depend only on the coordinates of the vertices of f.

[8] Again, to avoid clutter, we do not show v''_k in Fig. 7.

[9] These coefficients are, of course, different from the ones we use in Case 1.

Expression for $area(S_4)$**:**

Facet S_4 is a trapezoid with parallel sides $\overline{v_k v'_k}$ and $\overline{v_j v'_j}$, of lengths $z_k - h$ and $z_j - h$, respectively, and height $|\overline{v'_j v'_k}| = ((x_j - x_k)^2 + (y_j - y_k)^2)^{1/2}$. Therefore,

$$area(S_4) = \big(|\overline{v_k v'_k}| + |\overline{v_j v'_j}|\big) \cdot \frac{|\overline{v'_j v'_k}|}{2} = -|\overline{v'_j v'_k}|h + |\overline{v'_j v'_k}|\frac{z_j + z_k}{2}.$$

It follows that $area(S_4)$ can be written in the form

$$area(S_4) = a_f^{(4)}h^2 + b_f^{(4)}h + c_f^{(4)}, \tag{10}$$

where $a_f^{(4)} = 0$, and $b_f^{(4)}$ and $c_f^{(4)}$ depend only on the coordinates of the vertices of f.

Expression for $area(S_5)$**:**

Let f_H be the vertical projection of f on the plane H, with vertices v'_i, v'_j, and v'_k, corresponding to v_i, v_j, and v_k, respectively. (Note that v_{ij} and v_{ik} get projected to themselves.) Then

$$area(S_5) = area(f_H) - area\big(\langle v'_i, v_{ij}, v_{ik}\rangle\big).$$

Now $area(\langle v'_i, v_{ij}, v_{ik}\rangle)$ has already been computed in Case 1 (Eq. (5)). Moreover, $area(f_H) = (((\boldsymbol{v}_j - \boldsymbol{v}_i) \times (\boldsymbol{v}_k - \boldsymbol{v}_i)) \cdot \boldsymbol{d})/2$.

Since the former is quadratic in h and the latter is a constant, it follows that $area(S_5)$ is quadratic in h. Thus,

$$area(S_5) = a_f^{(5)}h^2 + b_f^{(5)}h + c_f^{(5)}, \tag{11}$$

for some coefficients $a_f^{(5)}$, $b_f^{(5)}$, and $c_f^{(5)}$ that depend only on the coordinates of the vertices of f. From Eqs. (7)–(11) and the fact that $\boldsymbol{Q}_5 \cdot \boldsymbol{N}_5 = -h$, we have

$$\begin{aligned}
active\text{-}volume_f(h) &= \frac{1}{3}\left|\sum_{j=1}^{5}(\boldsymbol{Q}_j \cdot \boldsymbol{N}_j)\big(a_f^{(j)}h^2 + b_f^{(j)}h + c_f^{(j)}\big)\right| \\
&= \frac{1}{3}\left|\sum_{j=1}^{4}(\boldsymbol{Q}_j \cdot \boldsymbol{N}_j)\big(a_f^{(j)}h^2 + b_f^{(j)}h + c_f^{(j)}\big) - h\big(a_f^{(5)}h^2 + b_f^{(5)}h + c_f^{(5)}\big)\right| \\
&= \frac{1}{3}\left|-a_f^{(5)}h^3 + \left(-b_f^{(5)} + \sum_{j=1}^{4}(\boldsymbol{Q}_j \cdot \boldsymbol{N}_j)a_f^{(j)}\right)h^2\right. \\
&\qquad \left. + \left(-c_f^{(5)} + \sum_{j=1}^{4}(\boldsymbol{Q}_j \cdot \boldsymbol{N}_j)b_f^{(j)}\right)h + \sum_{j=1}^{4}(\boldsymbol{Q}_j \cdot \boldsymbol{N}_j)c_f^{(j)}\right|.
\end{aligned}$$

Therefore,

$$active\text{-}volume_f(h) = a_f h^3 + b_f h^2 + c_f h + d_f, \tag{12}$$

for some coefficients a_f, b_f, c_f, and d_f that depend only on the coordinates of the vertices of f. The lemma now follows if we sum the active-volume contributions of all front and back facets, using Eqs. (6) and (12), respectively.

We now discuss how to handle the other cases mentioned at the beginning of the proof. Let H intersect f such that f^- is a quadrilateral and f^+ is a triangle. If f is a back facet, then we can use the discussion from Case 1 to compute the active-volume. Suppose that $z_i > z_j$, $z_i > z_k$, and the vertices v_i, v_j, and v_k are in counterclockwise order with respect to $\boldsymbol{n}_f$. All the formulas for the facet areas still apply. However, the directions of the normals $\boldsymbol{N}_j$ must now be reversed (except for $\boldsymbol{N}_2$). Also, now $\boldsymbol{Q}_4 \cdot \boldsymbol{N}_4 = -h$, and therefore the coefficients of $area(S_4)$ in the active-volume formula must be taken with opposite signs.
If f is a front facet we use the discussion from Case 2 to compute the active-volume. Again, suppose that $z_i > z_j$, $z_i > z_k$, and the vertices v_i, v_j, and v_k are given in counterclockwise order with respect to $\boldsymbol{n}_f$. All the formulas for the facet areas still apply. However, the directions of the normals $\boldsymbol{N}_j$ must now be reversed (except for $\boldsymbol{N}_2$). Also, now $\boldsymbol{Q}_5 \cdot \boldsymbol{N}_5 = h$, and therefore the coefficients of $area(S_5)$ in the active-volume formula must be taken with opposite signs. □

Next we obtain an expression for the inactive-volume.

Lemma 4.2. *Let the plane H be at height h above the xy-plane. The total support volume contributed by the inactive facets* (*i.e., the inactive-volume*) *is of the form $Ch + D$, where the coefficients C and D depend only on the coordinates of the vertices of the inactive facets.*

Proof. Let f be any inactive back facet with vertices $v_i = (x_i, y_i, z_i)$, $v_j = (x_j, y_j, z_j)$, and $v_k = (x_k, y_k, z_k)$, where $z_i \leqslant z_j \leqslant z_k$; see Fig. 8. (The other cases are similar and discussed at the end of

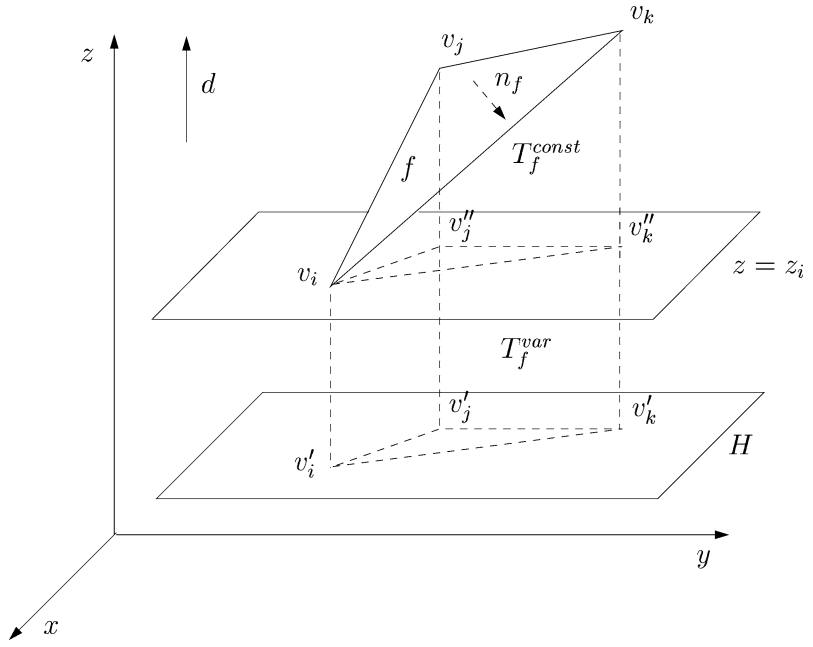

Fig. 8. The support polyhedron $T_f = T_f^{\text{const}} \cup T_f^{\text{var}}$ contributed by an inactive back facet f. T_f^{const} is defined by the vertices v_i, v_j, v_k, v_j'', and v_k''. T_f^{var} is defined by v_i, v_i', v_j', v_k', v_j'', and v_k''.

the proof.) We will prove that the support volume contributed by f is of the form $c_f h + d_f$ where the coefficients c_f and d_f depend only on the coordinates of the vertices of f. This implies the result.

Since f is an inactive back facet it will contribute to the support volume only if H is below f. Let v_j'' and v_k'' be the projections of v_j and v_k, respectively, on the plane passing through v_i and parallel to H. Let v_i', v_j', and v_k' be the projections of v_i, v_j, v_k, respectively, on H. We would like to find a formula for the volume of the polyhedron, T_f, defined by the vertices v_i, v_j, v_k, v_i', v_j' and v_k'.

T_f consists of two polyhedra. The first, denoted T_f^{const}, is defined by v_i, v_j, v_k, v_j'', and v_k'', and the second, denoted T_f^{var}, is defined by v_i, v_i', v_j', v_k', v_j'', and v_k''. As long as H is below f (i.e., f is inactive), the volume of T_f^{const} does not change when H is moved. The volume, $vol(T_f^{\text{const}})$, of T_f^{const} can, in fact, be computed using Eq. (12) in Lemma 4.1, evaluated at $h = z_i$.

The polyhedron T_f^{var} is a prism with base $\langle v_i, v_k'', v_j''\rangle$, and height $z_i - h$. The volume, $vol(T_f^{\text{var}})$, of T_f^{var} can be computed as

$$vol\left(T_f^{\text{var}}\right) = area\left(\left\langle v_i, v_k'', v_j''\right\rangle\right) \cdot (z_i - h),$$

where $area(\langle v_i, v_k'', v_j''\rangle)$ is just the area of the vertical projection of f onto the plane $z = z_i$, and depends only on the coordinates of the vertices of f. (It can also be computed using Eq. (11) in Lemma 4.1, evaluated at $h = z_i$.)

Let us denote by $inactive\text{-}volume_f(h)$ the support volume contributed by an inactive facet f, with respect to the plane $H\colon z = h$. Then

Table 5
Minimum support volume for convex polyhedra generated from random points on a sphere of radius 100. Here "non-decomp. volume" refers to the support volume when the polyhedra are built without decomposition; observe the significant reductions achieved via decomposition. Note that the support volume without decomposition is roughly equal to the volume under the lower hemisphere, as it should be

Support volume				
#verts n	min volume	$h_{\min}$	run time (s)	non-decomp. volume
20,000	2.7	−0.17	4.1	1,046,037.3
40,000	1.0	0.02	8.2	1,047,241.1
60,000	0.6	−0.08	12.6	1,047,256.4
80,000	0.3	−0.04	16.7	1,047,234.3
100,000	0.3	0	20.9	1,047,201.5
120,000	0.2	0.03	25.5	1,047,197.6
140,000	0.2	−0.02	29.7	1,047,167.5
160,000	0.1	0	34.1	1,047,192.4
180,000	0.1	0.02	37.9	1,047,188.5
200,000	0.1	−0.03	43.2	1,047,200.0

Table 6
Minimum support volume for convex polyhedra generated from random points on a cone; each polyhedron has been rotated by the indicated angle to make it non-symmetric about the origin. Here "non-decomp. volume" refers to the support volume when the polyhedra are built without decomposition; observe the significant reductions achieved via decomposition

Support volume					
#verts n	angle	min volume	$h_{\min}$	mean run time (s)	non-decomp. volume
20,001	200	10,999.5	−21.1	4.2	965,039.4
	37	26,389.1	21.5		1,107,383.9
40,001	40	24,003.9	21.2	8.6	947,591.6
	75	3,631.2	−4.4		994,452.0
60,001	240	12,900.6	−4.1	12.7	815,518.8
	112	8,118.6	2.4		865,432.8
80,001	80	1,359.2	−4.2	17.3	1,014,386.0
	150	20,249.8	−24.6		887,435.1
100,001	280	1,349.3	−4.2	21.5	1,014,432.7
	187	1,068.5	−10.7		1,033,711.3
120,001	120	12,892.3	−4.1	25.8	815,171.4
	225	13,784.8	−19.8		767,197.8
140,001	320	22,932.9	21.4	30.3	948,013.1
	263	549.0	3.2		959,299.2
160,001	160	10,760.1	−21.2	34.7	962,983.8
	300	12,908.9	4.0		849,211.7
180,001	0	0	1.7	39.5	2,128,798.5
	338	14,356.3	21.2		1,754,483.3
200,001	200	10,749.7	−21.2	43.6	962,390.9
	15	6,736.2	17.2		1,952,343.8

$$\begin{aligned} \textit{inactive-volume}_f(h) &= \textit{vol}\big(T_f^{\text{var}}\big) + \textit{vol}\big(T_f^{\text{const}}\big) \\ &= \textit{area}\big(\langle v_i, v_k'', v_j''\rangle\big) \cdot (z_i - h) + \textit{vol}\big(T_f^{\text{const}}\big) \\ &= c_f h + d_f, \end{aligned} \tag{13}$$

for some coefficients c_f and d_f that depend only on the coordinates of the vertices of f.

Finally, if f is a front facet, then a symmetric discussion applies.

The lemma follows by summing the volume contributions of all inactive facets using Eq. (13). □

4.1. The algorithm

The algorithm is similar to the one in Section 3.1. We sweep the plane H upwards, visiting the vertices in sorted order and at each vertex update the active-volume and inactive-volume terms based on the facets incident to the current vertex. The three cases outlined in the algorithm in Section 3.1 hold here as well, except that all references to the active-area and inactive-area terms are now to the active-volume and inactive-volume terms, respectively. At each vertex we minimize the sum of the active-volume and inactive-volume terms (hence the total support volume) as the position, h, of H ranges between the z-coordinates of the current vertex and the next one. This gives the optimal position for H between successive vertices. We repeat this for all vertices to find the globally-best position for H.

Theorem 4.1. *The support volume version of Problem* 2.1 *can be solved in* $O(n \log n)$ *time for a convex polyhedron* $\mathcal{P}$ *with* n *vertices* ($O(n)$ *time if the vertices are given in sorted order in the decomposition direction* $\boldsymbol{d}$).

Proof. Similar to the proof of Theorem 3.1. □

4.2. Experimental results

We have also implemented the above support volume minimization algorithm. The details of the implementation and the test polyhedra are the same as in Section 3.2. Tables 5 and 6 summarize our results.

5. Decomposing non-convex polyhedra

We assume hereafter that $\mathcal{P}$ is a non-convex polyhedron. Such polyhedra pose problems that are not encountered in the convex case. First, unlike the convex case, supports need not extend all the way to the platform, but instead may terminate at some other point on the polyhedron itself. Furthermore, it is possible that only part of a facet is in contact with supports, unlike the convex case where either the entire facet or no part of it is in contact with supports. Finally, it is possible that parallel facets are also in contact with supports, either fully or partially. Fortunately, despite these problems, it is possible to handle a non-convex polyhedron in a fashion similar to the convex case, after doing some initial processing on it.

For now, we ignore the issue of controlling the number of pieces in the decomposition; we discuss this in Section 5.7. Furthermore, for concreteness, we focus on the contact-area version of Problem 2.1. (In fact, the volume problem is easier, because parallel facets do not contribute to support volume – as they do to contact-area – and, therefore, can be essentially ignored.)

5.1. Overview of approach

The main idea is to partition each front or back facet of $\mathcal{P}$ into two classes of triangles, called black and gray triangles. (One of these classes may be empty.) A black triangle t will always be completely in contact with supports, regardless of the position of the decomposition plane H; therefore, it always contributes a fixed amount, $area(t)$, to the total contact-area and so can be ignored for minimization purposes. However, a gray triangle t will contribute anywhere between zero and $area(t)$ to the total contact-area, depending on the position of H, and so needs to be accounted for.

Parallel facets are treated a little differently. A parallel facet of $\mathcal{P}$ is partitioned into three classes of triangles, called black, gray, and white. (Up to two of these classes may be empty.) Black and gray triangles have the same interpretation as above. A white triangle will never be in contact with supports, regardless of the position of H, and so can be ignored for minimization purposes.

From the above discussion it is clear that only gray facets are relevant to the minimization problem. As we will see, these can be handled using the approach in Section 3.

5.2. Black, gray, and white triangles

We next define formally the different types of triangles and establish their stated properties.

Let f be a front facet. Imagine that we build $\mathcal{P}$, without decomposition, in direction $\boldsymbol{d}$. Consider the supports (if any) that are in contact with f. Their *footprint* on f, i.e., their intersection with f, is a collection of polygons, which we will call *black polygons*. We triangulate the black polygons of f to obtain the set, B_f, of *black triangles* associated with f. Let p be any point in some black triangle. Since p is in contact with supports for build direction $\boldsymbol{d}$, the ray r_p, emanating from p in this direction must intersect $\mathcal{P}$. If p' is the first intersection of r_p and $\mathcal{P}$ – not counting p – then the segment $\overline{p'p}$ is the support for p'. (Informally, p is in contact with supports because some part of $\mathcal{P}$, i.e., p', is directly above p in direction $\boldsymbol{d}$.) The complement of the black polygons on f is a collection of *gray polygons*, and the set, G_f, of *gray triangles* associated with f is obtained by triangulating the gray polygons. No point in a gray triangle is in contact with supports for build direction $\boldsymbol{d}$.

Now, suppose that we build $\mathcal{P}$ with decomposition. Let t be a black triangle in B_f. Assume that the decomposition plane H intersects t, and let t^- and t^+ be the parts of t above and below H, respectively. That is, $t^- \in \mathcal{P}^-$ and $t^+ \in \mathcal{P}^+$. Now, all of t^- will require support since it belongs to the front facet f, and $\mathcal{P}^-$ is built in direction $-\boldsymbol{d}$. The polyhedron $\mathcal{P}^+$ is built in direction $\boldsymbol{d}$, and, furthermore, the part of $\mathcal{P}^+$ that is directly above t^+ is the same as the part of $\mathcal{P}$ that would be directly above t^+ if $\mathcal{P}$ were built without decomposition in direction $\boldsymbol{d}$. Since t is a black triangle, t^+ is in contact with supports for direction $\boldsymbol{d}$ in the case without decomposition, and, hence, also in the case with decomposition. Thus, all of t is in contact with supports.

If H is completely above t, then $t^+ = \emptyset$ and $t^- = t$; if H is completely below t, then $t^- = \emptyset$ and $t^+ = t$. In either case, the above argument still applies. It follows that regardless of the position of H, if t is a black triangle, then all of it is in contact with supports. Thus, black triangles may be ignored.

On the other hand, suppose that t is a gray triangle in G_f, where f is a front facet. Assume that H intersects t. As in the case of a black triangle, the part $t^- \in \mathcal{P}^-$ will be in contact with supports. Consider the part $t^+ \in \mathcal{P}^+$, which is built in direction $\boldsymbol{d}$. No part of t^+ will be in contact with supports, since in the case where $\mathcal{P}$ is built in direction $\boldsymbol{d}$, without decomposition, no part of t is in contact with supports. If H is completely below or completely above t, then either t is not in contact with supports at all or

is completely in contact with supports. Therefore, it follows that t's contribution to the total contact-area depends on the position of H. In fact, this contribution is exactly the same as for a front facet in the convex case. That is, if t is inactive, then its contribution is zero or $area(t)$; if t is active, then its contribution is a quadratic function of the position, h, of H, as in Section 3.

So far, we have considered the case where the facet $f \in \mathcal{P}$ is a front facet. If f is a back facet of $\mathcal{P}$ then we define the black and gray triangles on f by building $\mathcal{P}$, without decomposition, in direction $-\boldsymbol{d}$. Then a discussion symmetric to the one above again shows that, in the case with decomposition, only gray triangles need be considered.

Finally, we turn to the case where f is a parallel facet. Suppose that we build $\mathcal{P}$ without decomposition in direction $\boldsymbol{d}$, and, independently, in direction $-\boldsymbol{d}$. The *black polygons* consist of all points on f that are in contact with supports for both directions. The *gray polygons* consist of all points on f that are in contact with supports for exactly one of the two directions. The *white polygons* consist of all points on f that are not in contact with supports for either of the two directions. By definition, these polygons partition f. The set B_f (respectively G_f, W_f) of *black* (respectively gray, white) triangles on f is obtained by triangulating the black (respectively *gray*, *white*) polygons.

Now suppose that we build $\mathcal{P}$ with decomposition. It is easy to see that regardless of the position of the decomposition plane H, a black triangle will always be in contact with supports, while a white triangle will never be in contact with supports. Therefore, both these types of triangles may be ignored. However, a gray triangle will contribute to the contact-area an amount which depends on the position of H. For instance, let $t \in G_f$ be a gray triangle which exists because f is in contact with supports when $\mathcal{P}$ is built, without decomposition, in direction $\boldsymbol{d}$. Then, in the case with decomposition, t^+ is in contact with supports and t^- is not. So the contribution of t to the total contact-area varies quadratically with the position, h, of H, from zero to $area(t)$. So, again only gray triangles need to be considered.

5.3. *Computing black and gray triangles for front and back facets*

We discuss how to to compute the footprint of the supports on each front facet, hence the black and gray triangles on it.

We will use a technique called *cylindrical decomposition* [29]. From each edge, e, of each back facet, b, we erect a strip, $V_{e,b}$, which passes exactly through e and extends vertically downwards, in direction $-\boldsymbol{d}$. As soon as a part of $V_{e,b}$ intersects another facet of $\mathcal{P}$ (which must be a front facet), we stop propagating that part below the intersected facet; however, we continue propagating the remaining parts of $V_{e,b}$.[10] Each such intersection of $V_{e,b}$ with a front facet will be a part of the footprints that we are trying to compute.

To perform this step efficiently, we compute the intersection of each front facet with $V_{e,b}$ to get a set, L, of line segments. (Since the facets of $\mathcal{P}$ do not intersect each other, the segments in L are non-crossing, but possibly touching.) We do a trapezoidal decomposition [13] of $L \cup \{e\}$ in the plane of $V_{e,b}$. We identify each trapezoid in this decomposition that is adjacent to e at the top and to a line segment $\ell \in L$ at the bottom. The bottom edge of this trapezoid is one of the sought intersections of $V_{e,b}$ with a front facet. We store this edge with the front facet that generated ℓ.

However, not all footprints on front facets will be discovered by the above process. For instance, if the projection of b completely covers a front facet, f, below it, then none of the strips $V_{e,b}$ erected from b

[10] The process resembles water cascading down from e.

will intersect f, and, yet, supports for b will rest on f. To handle such situations, we also erect from each edge, e, of each front facet, f, a strip $V_{e,f}$ vertically upwards, stopping the propagation of any part of the strip as soon as it intersects a (back) facet above it, while continuing to propagate other parts. To compute these intersections we do a trapezoidal decomposition of the set $L' \cup \{e\}$, where L' contains the intersections of all the back facets with $V_{e,f}$. For each trapezoid in this decomposition that is incident to e at the bottom and to a segment $\ell' \in L'$ at the top, we take the top edge of the trapezoid as the intersection of $V_{e,f}$ with the back facet that generated ℓ', and store it with that back facet.

After we have carried out the above steps for all front and back facets, we have associated with each facet a list of line segments corresponding to intersections of the different vertical strips with the facet. Since a strip is not propagated below an intersected facet, it is easy to see that the line segments associated with a facet are non-crossing (but may be touching). For each facet, we compute the arrangement [13] of the set consisting of the associated segments and the edges bounding the facet.

Let f be any front facet and let c be any cell of the arrangement computed on f. Then c is the footprint of a support on f for build direction $\boldsymbol{d}$, hence a black polygon, iff there is a cell c' on a back facet b above f, such that c' projects exactly to c. (The cells c and c' form the bottom and top facets of a support cylinder; the other facets of this cylinder are vertical and bounded below and above by edges of c and c'.) Any other cell of f is a gray polygon.

We can identify the black triangles of f directly (instead of first computing the black polygons). We project the arrangement on each front facet to the xy-plane, triangulate its cells, and then lift the triangles back to the front facet. We make a list, F, of these lifted triangles along with their centroids, and sort F lexicographically on the x-, y-, and z-coordinates of the centroids, taken in that order. We make a similar sorted list B for the the back facets. Note that if a cell, c, from a front facet and a cell, c', from a back facet form the bottom and top facets of a support cylinder, then c and c' have identical projections on the xy-plane and will, therefore, be triangulated identically (if a deterministic triangulation algorithm is used, e.g., [13, Chapter 3]). It should be clear now that a simultaneous scan of the two sorted lists suffices to identify matching pairs of triangles, where one triangle is from B and the other is from F such that the former is above the latter and projects exactly to it. For each matching pair, the triangle from F is a black triangle on some front facet; all unmatched triangles of F are gray triangles on front facets. (This approach for matching triangles, using a lexicographic sort, is due to [11].)

In this way, we can compute all the black and gray triangles for each front facet. A symmetric approach yields the black and gray triangles for the back facets.

Lemma 5.1. *The set of black and gray triangles for all front and back facets of an n-vertex polyhedron $\mathcal{P}$ can be computed in $O(n^2 \log n)$ time.*

Proof. Correctness is clear from the discussion above. We establish the running time.

The number of line segments generated by intersecting front facets with any strip $V_{e,b}$ is $O(n)$, so their trapezoidal decomposition has size $O(n)$ and can be computed in time $O(n \log n)$ per strip, hence $O(n^2 \log n)$ for all strips.

The total number of segments (intersections) created on all the front facets of $\mathcal{P}$ by the propagation of any $V_{e,b}$ is $O(n)$. This is because the trapezoidal decomposition has $O(n)$ trapezoids and the segments of interest are the bottom edges of a subset of these trapezoids. Since there are $O(n)$ $V_{e,b}$'s, the total number of segments generated by them on all front facets is $O(n^2)$.

Now, consider a front facet, f, and let m_f be the total number of segments on f taken over all strips $V_{e,b}$. By the above discussion, $\sum_f m_f = O(n^2)$. (Note that m_f could be $\Theta(n^2)$ for an individual facet f. This could happen if, for instance, $\Theta(n)$ back facets are above f and overlap in the form of a trellis. However, $\sum_f m_f$ is still $O(n^2)$.) As noted earlier in the discussion of the algorithm, since a strip is not propagated below an intersected facet, the segments on f are non-crossing; thus, their arrangement has complexity $O(m_f)$. It can be computed and triangulated in time $O(m_f \log m_f)$, and yields $O(m_f)$ triangles. Taken over all front facets, there are $O(\sum_f m_f) = O(n^2)$ such triangles, and they can be generated and sorted in time $O(\sum_f m_f \log m_f) = O(n^2 \log n)$. Likewise for the triangles from all the back facets. The resulting lists of $O(n^2)$ triangles each can be scanned in $O(n^2)$ additional time to determine the black and gray triangles. □

5.4. Computing black, gray, and white triangles for parallel facets

Let f be a parallel facet. Recall that a black (respectively gray, white) triangle on f is one which is in contact with supports for both (respectively exactly one, neither) of the directions $\boldsymbol{d}$ and $-\boldsymbol{d}$. Let V_f be the vertical strip which is in the plane of f and exactly contains it. We may assume without lost of generality that no vertex of f is in the interior of V_f. (Each bounding line of V_f contains at least one vertex of f. If there is a vertex in the interior of V_f, we draw a vertical line through it and split f into two facets that each satisfy the assumption.) Let vertex u of f lie on one of the bounding lines of V_f and let vertices v and w lie on the other, with v above w in direction $\boldsymbol{d}$; note that the line segments $\overline{uv}$ and $\overline{uw}$ span V_f. Consider the back facets of $\mathcal{P}$ that either pierce V_f above $\overline{uv}$, or touch V_f above $\overline{uv}$ and are in the same halfspace of V_f as the outer unit-normal, $\boldsymbol{n}_f$, of f. (These are the back facets whose supports are potentially in contact with f when $\mathcal{P}$ is built in direction $\boldsymbol{d}$.) The intersections of these back facets with V_f is a set, A, of line segments. We do a trapezoidal decomposition of A in the plane of V_f (note that we do not consider the edges of f when doing the decomposition). Let T be the set of trapezoids in this decomposition that are bounded from above by some segment of A and are unbounded below. Let T_f be the trapezoids obtained by intersecting the ones in T with f. (Strictly speaking, some of the elements in T_f may be triangles, which we take to be degenerate trapezoids.)

Fig. 9 illustrates the computation of T_f. Fig. 9(a) shows the back facets, b_1, b_2, b_3, and b_4, that pierce or touch V_f above $\overline{uv}$: b_1 and b_4 pierce V_f, b_2 touches V_f and is in the same halfspace of V_f as $\boldsymbol{n}_f$, and b_3 touches V_f but is in the halfspace of V_f that does not contain $\boldsymbol{n}_f$. The supports needed by b_3 will not be in contact with f, so we ignore it. The set A consists of the line segments a_1, a_2, and a_4. Fig. 9(b) shows the trapezoidal decomposition of A. The trapezoids of T are t_1, t_2, and t_4; their intersections with f, shown shaded, yields the set T_f.

Symmetrically, we consider the front facets of $\mathcal{P}$ that intersect V_f at or below $\overline{uw}$. These intersections yield a set, A', of line segments. We do a trapezoidal decomposition of A', take the set T' of trapezoids in this decomposition that are bounded from below by some segment of A' and are unbounded above, and intersect these with f to get a set T'_f of trapezoids.

The set of black polygons for f is obtained by taking the intersections of every pair of trapezoids, one in T_f and the other in T'_f. Any part of a trapezoid in T_f or T'_f that is not a black polygon is a gray polygon. The complement of the union of the gray and black polygons on f is the set of white polygons. We can compute these three classes of polygons easily by sorting the vertical edges in T_f and T'_f into two lists and then doing a simultaneous scan of the two lists. Since these polygons are actually trapezoids, we can generate the corresponding triangles easily.

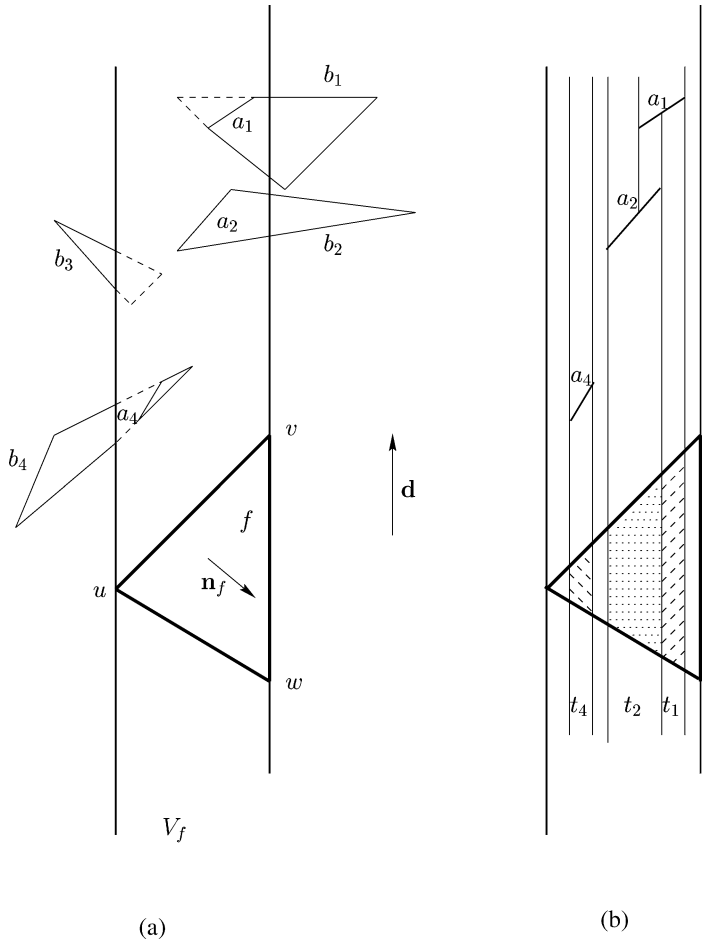

Fig. 9. Computation of the set T_f (shown shaded) for a parallel facet f.

Lemma 5.2. *The set of black, gray, and white triangles for all parallel facets of an n-vertex polyhedron $\mathcal{P}$ can be computed in $O(n^2 \log n)$ time.*

Proof. The correctness of the method is clear from the above discussion. For the running time, note that the sets A and A' each have size $O(n)$ and can be computed in $O(n)$ time. Doing the trapezoidal decomposition and computing T_f and T'_f takes $O(n \log n)$ time. Finally, sorting and scanning the lists and computing the different polygons takes $O(n \log n)$ time. Therefore each of the $O(n)$ parallel facets can be handled in time $O(n \log n)$, and the lemma follows. □

5.5. *Decomposing to minimize contact-area*

At this point, we have for each front, back and parallel facet of $\mathcal{P}$, a list of the black, gray, and white polygons. As discussed in Section 5.2, only the gray triangles are relevant when decomposing $\mathcal{P}$ to minimize the contact-area of supports. We store the subdivision defined by the union of the set of gray

triangles in a doubly-connected edge list and perform a sweep over them as in Section 3 to compute the optimum decomposition plane H. (Even though the algorithm in Section 3 is for convex polyhedra, the sweep does not depend on convexity per se, and so it works for our collection of gray triangles as well.)

As seen above, the set of gray triangles can be computed in time $O(n^2 \log n)$. Also, from the proofs of Lemmas 5.1 and 5.2, the number of gray triangles is $O(n^2)$. The time to sort the vertices of the induced subdivision by z coordinates, in preparation for the sweep, is $O(n^2 \log n)$. During the sweep, each vertex can be processed in time proportional to its degree, which implies a time bound of $O(n^2)$ for processing all vertices, since the subdivision has $O(n^2)$ edges.

As noted at the beginning of Section 5, the volume minimization problem is easier, and essentially the same approach works for this also. We will see in Section 5.7 that the size of the decomposition can be controlled in $O(n \log n)$ time. We conclude:

Theorem 5.1. *The contact-area and support volume versions of Problem* 2.1 *can be solved in* $O(n^2 \log n)$ *time for a non-convex polyhedron* $\mathcal{P}$ *with* n *vertices.*

5.6. Experimental results

Our primary goal was to investigate the extent of support reduction achievable via decomposition of typical models. Therefore, for convenience, we implemented a simpler, but slower, version of the algorithm described above, for support volume minimization. The algorithm differs mainly in how the footprints are computed – instead of using the earlier algorithm for cylindrical decomposition, we use a somewhat less efficient approach, as follows.

Let f be a fixed front facet and let b be any back facet. We project f and b to the xy-plane and compute the intersection of their projections (i.e., triangles), which yields a convex polygon, $C(b)$. If $C(b) \neq \emptyset$, then let p be any point in it, say the centroid. If the pre-images, p_f and p_b, of p on f and b, respectively, are such that p_b is above p_f in direction $\boldsymbol{d}$, then p_f is in contact with supports. This implies that the pre-image $C_f(b)$ of $C(b)$ on f is in contact with supports. This follows since no facet of $\mathcal{P}$ pierces another, so there cannot be another point p' in $C(b)$, whose pre-images on f and b are in the opposite order from those of p. (Note that it need not be the case that the cylinder bounded by $C_f(b)$ and by $C_b(b)$ – the pre-image of $C(b)$ on b – is a support cylinder, since b, or parts thereof, need not be immediately above f; there could be parts of other back facets in between.) Given the polygons $C(b)$ that are found to be in contact with supports, we can compute the footprint of supports on f by taking the union of the pre-images, $C_f(b)$, of these polygons. (In our implementation, we used the functions provided by LEDA [28] to perform the union and intersection operations.) We triangulate the footprint (respectively the complement of the footprint) on f to get the black (respectively gray) triangles on f. We handle back facets in a symmetric fashion, using direction $-\boldsymbol{d}$ instead of $\boldsymbol{d}$. Thereafter, we apply the sweep-based algorithm on the set of all gray triangles to compute the optimum position of the decomposition plane.

The most expensive part of this algorithm turns out to be the union step in the computation of the footprints. Note that the algorithm simply projects all back facets down to the xy-plane, without regard to any intervening facets. Thus, the complexity of the union of the polygons $C(b)$ on a single front facet, f, can be $\Theta(n^2)$ in the worst case, and $\Theta(n^3)$ over all front facets. (An example of this is a configuration of $\Theta(n)$ front facets stacked on top of each other and $\Theta(n)$ back facets above them that overlap in the

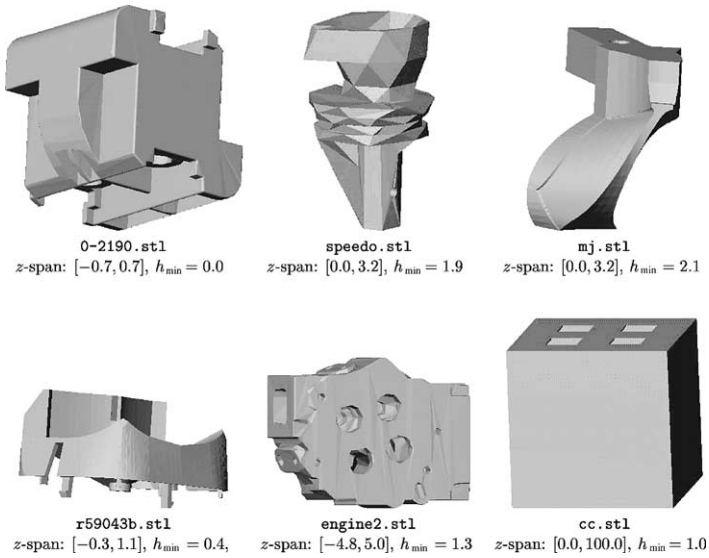

Fig. 10. Non-convex models tested. Here "z-span" denotes the span of coordinates in the z-direction; it is given to help visualize the position, $h_{\min}$, of the decomposition plane. The decomposition direction, $\boldsymbol{d}$, points vertically upwards.

form of a trellis. Notice that this situation would not create a problem for cylindrical decomposition, since there we do not project through intervening front facets.) The total time to compute the union is $O(n^2 \log n)$ in the worst case for any front facet [13], hence $O(n^3 \log n)$ over all front facets.

We performed our experiments on the six models shown in Fig. 10. The first five models were obtained from Stratasys, Inc., a Minnesota-based company specializing in LM. To keep the running time of our algorithm on these models reasonable, we reduced the number of facets in some of the larger models – specifically, `0-2190.stl` (to about 25% of its original number), `speedo.stl` (to about 15%), and `engine2.stl` (to about 18%). For this we used the *Decimator* software package from Raindrop Geomagic, Inc., a North Carolina-based company specializing in the design of CAD software. This package reduces model sizes while preserving the original topology to the maximum extent possible. The sixth model, `cc.stl`, was hand-generated; it is used to show that on some models our algorithm may not achieve any reduction in support requirements.

Table 7 summarizes our experimental results for the six models. To give an idea of the relative sizes of the models, the table also gives the dimensions of their axes-parallel bounding boxes. The models are listed in the table in decreasing order of the reduction in support volume that is achieved (the sixth column divided by the fourth). On the first four models, our algorithm achieved a reduction ranging from a factor of seven to about a factor of four. On the fifth model, it achieved a reduction of about 1.4. It achieved no reduction at all on the last model, `cc.stl`, which is essentially a hollow cube, with walls of thickness 1 and with four holes on the top face. If this model is built without decomposition, the interior of the cube will be filled with supports, except for the regions directly below the holes. Any decomposition plane that intersects the model strictly above the inner base (which is at height 1), will generate two pieces for which the total support volume is higher than without decomposition, since the region in the lower piece that was originally below the holes is now filled with supports. The optimal position for decomposition is any height in the interval $[0, 1]$.

Table 7
Experimental results for non-convex models. Experiments were performed on a SUN Ultra 60 Sparc machine with 512 MB of main memory and a 450 MHz processor

	model	# facets	min volume	$h_{\min}$	non-decomp. volume	run time (s)	bounding box $l \times w \times h$
1	`0-2190.stl`	3,492	0.1	0.0	0.7	16,661	$1.4 \times 1.5 \times 1.4$
2	`speedo.stl`	2,500	0.9	1.9	5.1	15,730	$2.0 \times 1.9 \times 3.2$
3	`mj.stl`	2,832	1.7	2.1	8.1	13,911	$2.8 \times 4.6 \times 3.2$
4	`r59043b.stl`	3,386	0.8	0.4	3.0	20,833	$2.4 \times 2.1 \times 1.4$
5	`engine2.stl`	4,180	174.9	1.3	251.6	41,156	$12.1 \times 6.2 \times 9.7$
6	`cc.stl`	112	823,210.0	1.0	823,210.0	4.7	$102.0 \times 102.0 \times 100.0$

5.7. Controlling the size of the decomposition

The optimal plane computed by the algorithm in Section 5.5 could decompose a non-convex polyhedron $\mathcal{P}$ into many polyhedra (as many as $\Theta(n)$ in the worst case), which is undesirable since it increases the cost of re-assembling $\mathcal{P}$. Ideally, the designer should be able to specify an integer K, and the algorithm should compute, among along all possible planes that generate no more than K polyhedra, a plane which is optimal with respect to support contact-area or volume. We show how this can be done by incorporating a preprocessing step in the algorithm. The idea is to partition the z-axis into $O(n)$ intervals, I_j, such that all planes whose heights are in I_j decompose $\mathcal{P}$ into the same number, k_j, of polyhedra. We then run the sweep algorithm of the previous section but do the minimization step only in those intervals I_j for which $k_j \leqslant K$.

Let $z_1 < z_2 < \cdots < z_t$, $t \leqslant n$, be the distinct z-coordinates of the n vertices of $\mathcal{P}$. The preprocessing involves two sweeps. The first sweep is in the positive z direction and it computes a set of intervals on the z-axis and associates with each interval an integer which is the number of connected components of $\mathcal{P}^-$ generated by any plane whose height is in the interval. Observe that the number of connected components of $\mathcal{P}^-$ with respect to a plane of height z_j is the same as the number with respect to a plane whose height is anywhere in the interval (z_j, z_{j+1}); let this number be k_j^-. Thus, the first sweep computes intervals of the form $[z_j, z_{j+1})$ and associates with each the integer k_j^-. Symmetrically, the second sweep is in the negative z direction and it computes intervals of the form $(z_j, z_{j+1}]$ and associates with each an integer k_j^+, which is the number of connected components of $\mathcal{P}^+$ with respect to any plane whose height is in $(z_j, z_{j+1}]$. Once these two sets of intervals have been computed, a single scan of them suffices to compute the desired intervals I_j and the corresponding integers k_j. Specifically, each interval I_j is either of the form $[z_j, z_j]$, with $k_j = k_j^- + k_{j-1}^+$, or of the form (z_j, z_{j+1}), with $k_j = k_j^- + k_j^+$.

We describe the first sweep in more detail. At any time, the vertices of the different connected components of $\mathcal{P}^-$ form a collection of disjoint sets. We maintain these using a Union-Find-Makeset data structure [12]. We initialize the structure to empty and set the current number, c, of connected components of $\mathcal{P}^-$ to zero. Let z_j be the current z-coordinate in the sweep and let V_j be the set of vertices of $\mathcal{P}$ at this z-coordinate. We consider each vertex $v \in V_j$ in turn and process it as follows. We create a new set containing just v and increment c by one. Then for each neighbor, u, of v such that u

is already in the Union-Find-Makeset data structure we do the following. If u and v are in different connected components, then we union the sets containing u and v, and decrement c by one. Notice that in this sweep, we only "add" edges to $\mathcal{P}^-$, so the connected components of $\mathcal{P}^-$ always merge, never split. Thus, a Union-Find-Makeset structure suffices to maintain the connected components of $\mathcal{P}^-$. After all vertices of V_j have been processed, we set k_j^- to c. At the end of the sweep, all the intervals $[z_j, z_{j+1})$, and their associated integers k_j^- will have been computed.

Excluding the $O(n \log n)$ time for the sorting, the algorithm takes $O(n\alpha(n))$ time, where $\alpha(n)$ is the slow-growing inverse Ackerman function. Hence this preprocessing step does not affect the asymptotic running time of the decomposition algorithm of Section 5.5.

6. Conclusions and future work

In this paper, we have presented a new approach to LM, in which the model is decomposed by a plane into several pieces that can be built independently and then glued together. Several advantages of this approach have been identified, including improved speed, greater versatility, and reduced support requirements. We have presented efficient geometric algorithms to decompose both convex and non-convex models so as to minimize the contact-area and the volume of the supports (Problem 2.1). For convex models, the algorithms are based on the plane sweep paradigm, where formulas for support contact-area and volume are generated, updated, and optimized as a function of the height of the sweep plane. Non-convex models are handled by first identifying certain critical facets that are relevant to the minimization and then using the same approach as in the convex case. Experimental results have been given that demonstrate the utility of the decomposition-based method.

There are actually four versions of Problem 2.1 depending on how we choose to build $\mathcal{P}^+$ and $\mathcal{P}^-$: (1) $\mathcal{P}^+$ and $\mathcal{P}^-$ in directions $\boldsymbol{d}$ and $-\boldsymbol{d}$, respectively – the version solved in the paper; (2) $\mathcal{P}^+$ and $\mathcal{P}^-$ in directions $-\boldsymbol{d}$ and $\boldsymbol{d}$, respectively; (3) $\mathcal{P}^+$ and $\mathcal{P}^-$ in directions $\boldsymbol{d}$ and $\boldsymbol{d}$, respectively; and (4) $\mathcal{P}^+$ and $\mathcal{P}^-$ in directions $-\boldsymbol{d}$ and $-\boldsymbol{d}$, respectively. For a given model, $\mathcal{P}$, the support requirements of one of the methods might be better than the other three. For instance, let $\mathcal{P}$ consist of two pyramids, with a common vertex v (i.e., one pyramid is a reflection of the other through v), and let $\boldsymbol{d}$ be perpendicular to the base of one pyramid. If we choose the decomposition plane H such that it contains v, then method 2 would require no supports. For any choice of H, the other three methods would require a non-zero amount of supports. Given a model, one strategy would be to run all four methods independently and pick the one which minimizes the support requirements.

For convex $\mathcal{P}$, the solutions for methods 2–4 are similar to that for method 1. For contact-area minimization, method 2 is completely symmetric to method 1. In method 3 (respectively method 4), only back (respectively front) facets need supports, regardless of the position of H; thus, the support contact-area equals the total area of the back (respectively front) facets, so any position of H is optimal. The situation is similar for support volume minimization, except that the platform for $\mathcal{P}^+$ and/or $\mathcal{P}^-$ need no longer be H. For instance, in method 2, the platform for $\mathcal{P}^+$ (respectively $\mathcal{P}^-$) is the plane parallel to H and coinciding with the extreme vertex in direction $\boldsymbol{d}$ (respectively $-\boldsymbol{d}$), so the support volume needs to be computed with respect to this platform rather than with respect to H. Similar statements apply to methods 3 and 4. In these methods, even though only back or front facets need supports, the position of H does affect the support volume (unlike contact-area); the optimal position for H can be found by minimizing the expression for the total support volume, as in method 1.

Unfortunately, the approach in Section 5 (where facets are decomposed into black, gray, and white triangles) does not extend to methods 2–4 when $\mathcal{P}$ is non-convex. Consider, for instance, method 2. Let t be a black triangle from a front facet and assume that H intersects t. Now, t^+ is completely in contact with supports, since it is built in direction $-\boldsymbol{d}$. However, the structure of the supports that will be in contact with t^- depends on the (yet-to-be-built) portion of $\mathcal{P}^-$ that is above H, which can be quite complex. (This problem did not arise in method 1, for reasons discussed at the beginning of Section 5.2.) Similar problems arise in method 3 for a black triangle belonging to a front facet and in method 4 for a black triangle belonging to a back facet. At present, we do not know how to overcome these difficulties and it appears that a different approach will be needed.

Throughout the paper, we have assumed a fixed decomposition direction $\boldsymbol{d}$ and found an optimum plane that is normal to $\boldsymbol{d}$. A challenging next step is to consider the problem of computing, over all directions $\boldsymbol{d}$, an optimum decomposition plane (while also limiting the number of pieces). The methods developed in this paper could be useful in computing such a plane once a small set of candidate directions has been identified.

Acknowledgements

We thank Stratasys, Inc. for providing us with STL files for our experiments. We also thank the referees for helpful comments that improved the paper substantially.

References

[1] P. Agarwal, P. Desikan, Approximation algorithms for layered manufacturing, in: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, 2000, pp. 528–537.
[2] S. Allen, D. Dutta, Determination and evaluation of support structures in layered manufacturing, J. Design Manufacturing 5 (1995) 153–162.
[3] B. Asberg, G. Blanco, P. Bose, J. Garcia-Lopez, M. Overmars, G. Toussaint, G. Wilfong, B. Zhu, Feasibility of design in stereolithography, Algorithmica 19 (1997) 61–83.
[4] M. Bablani, A. Bagchi, Quantification of errors in rapid prototyping processes and determination of preferred orientation of parts, in: Transactions of the 23rd North American Manufacturing Research Conference, 1995.
[5] M.J. Bailey, Tele-manufacturing: Rapid prototyping on the Internet, IEEE Comput. Graphics Appl. 15 (6) (1995) 20–26.
[6] G. Barequet, Y. Kaplan, A data front-end for layered manufacturing, Comput. Aided Design 30 (4) (1998) 231–243.
[7] B.G. Baumgart, A polyhedron representation for computer vision, in: Proceedings of the AFIPS National Computer Conference, Vol. 44, 1975, pp. 589–596.
[8] J.H. Bøhn, Removing zero-volume parts from CAD models for layered manufacturing, IEEE Comput. Graphics Appl. 15 (6) (1995) 27–34.
[9] P. Bose, Geometric and computational aspects of manufacturing processes, Ph.D. Thesis, School of Computer Science, McGill University, Montréal, Canada, 1995.
[10] C. Bradford Barber, D.P. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, ACM Trans. Math. Software 22 (4) (1996) 469–483, see also http://www.geom.umn.edu/software/qhull/.
[11] B. Chazelle, L. Palios, Triangulating a nonconvex polytope, Discrete Comput. Geom. 5 (1990) 505–526.
[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press and McGraw-Hill, 1990.
[13] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational Geometry: Algorithms and Applications, Springer, Berlin, 1997.
[14] A. Dolenc, I. Mäkelä, Slicing procedures for layered manufacturing techniques, Comput. Aided Design 26 (1994) 119–126.

[15] D. Dutta, V. Kumar, M. Pratt, R. Sriram, Towards STEP-based data transfer in Layered Manufacturing, in: Proceedings of the Tenth International IFIP WG 5.2/5.3 Conference PROLOMAT, 1998, see also http://www.mel.nist.gov/msidlibrary/summary/9826.html.
[16] S. Fekete, J. Mitchell, Histogram decomposition and stereolithography, 1997, Manuscript, http://www.zpr.uni-koeln.de/∼paper/paper.php3?paper=280.
[17] D. Frank, G. Fadel, Preferred direction of build for rapid prototyping processes, in: Proceedings of the 5th International Conference on Rapid Prototyping, 1994, pp. 191–200.
[18] R.N. Goldman, Area of planar polygons and volume of polyhedra, in: J. Arvo (Ed.), Graphics Gems II, Academic Press, 1991, pp. 170–171.
[19] P. Jacobs, Rapid Prototyping and Manufacturing: Fundamentals of StereoLithography, McGraw-Hill, 1992.
[20] E. Johnson, Support generation for three-dimensional layered manufacturing, Master's Project Report, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, 1999.
[21] P. Kulkarni, D. Dutta, An accurate slicing procedure for layered manufacturing, Comput. Aided Design 28 (1996) 683–697.
[22] J. Majhi, Geometric methods in computer-aided design and manufacturing, Ph.D. Thesis, Department of Computer Science and Engineering University of Minnesota, Minneapolis, MN, 1998.
[23] J. Majhi, R. Janardan, J. Schwerdt, M. Smid, Multi-criteria optimization algorithms for layered manufacturing, in: Proceedings of the 14th Annual ACM Symposium on Computational Geometry, 1998, pp. 19–28.
[24] J. Majhi, R. Janardan, J. Schwerdt, M. Smid, P. Gupta, Minimizing support structures and trapped area in two-dimensional layered manufacturing, Computational Geometry 12 (1999) 241–267.
[25] J. Majhi, R. Janardan, M. Smid, P. Gupta, On some geometric optimization problems in layered manufacturing, Computational Geometry 12 (1999) 219–239.
[26] A. Marsan, V. Kumar, D. Dutta, M. Pratt, An assessment of data requirements and data transfer formats for Layered Manufacturing, Technical Report NISTIR 6216, National Institute of Standards and Technology, September 1998, see also http://www.mel.nist.gov/msidlibrary/summary/9821.html.
[27] S. McMains, C. Séquin, A coherent sweep plane slicer for layered manufacturing, in: Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications, 1999, pp. 285–295.
[28] K. Mehlhorn, S. Näher, LEDA: A Platform for Combinatorial and Geometric Computing, Cambridge University Press, Cambridge, UK, 1999.
[29] K. Mulmuley, Computational Geometry: An Introduction through Randomized Algorithms, Prentice-Hall, 1993.
[30] J. Schwerdt, M. Smid, R. Janardan, E. Johnson, J. Majhi, Protecting critical facets in layered manufacturing, Computational Geometry 16 (2000) 187–210.
[31] J. Schwerdt, M. Smid, J. Majhi, R. Janardan, Computing the width of a three-dimensional point-set: an experimental study, ACM J. Experimental Algorithmics 4 (8) (1999), available online at http://www.jea.acm.org/1999/SchwerdtWidth/.
[32] P. Stucki, J. Bresenham, R. Earnshaw, Computer graphics in rapid prototyping technology, IEEE Comput. Graphics Appl. 15 (6) (1995) 17–19, Guest Editors' introduction.